# 27th IFIP WG 1.5 International Workshop on Cellular Automata and Discrete Complex Systems

**AUTOMATA 2021, July 12–14, 2021,**
**Aix-Marseille University, France**

Edited by

Alonso Castillo-Ramirez
Pierre Guillon
Kévin Perrot


OASIcs – Vol. 90 – AUTOMATA 2021 www.dagstuhl.de/oasics

*Editors*

**Alonso Castillo-Ramirez**
Universidad de Guadalajara, Mexico
alonso.castillor@academicos.udg.mx

**Pierre Guillon**
CNRS & Université d'Aix-Marseille, France
pierre.guillon@math.cnrs.fr

**Kévin Perrot**
Université d'Aix-Marseille, France
kevin.perrot@lis-lab.fr

*ACM Classification 2012*
Theory of computation → Models of computation; Mathematics of computing; Computing methodologies

**ISBN 978-3-95977-189-4**

*Published online and open access by*
Schloss Dagstuhl – Leibniz-Zentrum für Informatik GmbH, Dagstuhl Publishing, Saarbrücken/Wadern, Germany. Online available at https://www.dagstuhl.de/dagpub/978-3-95977-189-4.

*Publication date*
June, 2021

*Bibliographic information published by the Deutsche Nationalbibliothek*
The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data are available in the Internet at https://portal.dnb.de.



Digital Object Identifier: 10.4230/OASIcs.AUTOMATA.2021.0

**ISBN 978-3-95977-189-4** **ISSN 1868-8969** **https://www.dagstuhl.de/oasics**

## OASIcs – OpenAccess Series in Informatics

OASIcs is a series of high-quality conference proceedings across all fields in informatics. OASIcs volumes are published according to the principle of Open Access, i.e., they are available online and free of charge.



**ISSN 1868-8969**

**https://www.dagstuhl.de/oasics**

# Contents



27th IFIP WG 1.5 International Workshop on Cellular Automata and Discrete Complex Systems (AUTOMATA 2021).
Editors: Alonso Castillo-Ramirez, Pierre Guillon, and Kévin Perrot
OpenAccess Series in Informatics
OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

# Preface

This volume contains the full papers presented at the 27$^{\text{th}}$ International Workshop on Cellular Automata (CA) and Discrete Complex Systems (DCS), AUTOMATA 2021.

AUTOMATA is an annual series established in 1995 as a collaboration forum between researchers in CA and DCS, and as the official annual event of the International Federation for Information Processing (IFIP), Technical Committee 1, on Foundations of Computer Science, Working Group 5, on Cellular Automata and Discrete Complex Systems. Current topics of the conference include, but are not limited to, dynamical, topological, ergodic and algebraic aspects of CA and DCS, algorithmic and complexity issues, emergent properties, formal languages, symbolic dynamics, tilings, models of parallelism and distributed systems, timing schemes, synchronous versus asynchronous models, phenomenological descriptions, scientific modeling, and practical applications.

This year's event takes place on July 12$^{\text{th}}$-14$^{\text{th}}$, 2021, in a hybrid mode, both online and at the *Centre International de Rencontres Mathématiques* (CIRM) in Marseille, France. It is colocated with the Workshop on Automata Networks, and both events are the opportunity to jointly celebrate Eric Goles's 70th birthday, from Universidad Adolfo Ibañez, Chile, who is a prominent actor in both communities. More information about the conference can be found at

`https://automata2021.lis-lab.fr/`.

This volume includes four papers written by the the invited speakers of the conference:

- Maximilien Gadouleau (Durham University, UK);
- Luca Priogioniero (Università degli Studi di Milano, Italy);
- Siamak Taati (American University of Beirut, Lebanon);
- Ilkka Törmä (University of Turku, Finland).

We sincerely thank them for accepting the invitation and their very valuable contributions.

We received nine submissions as full papers to the conference. Each submission was reviewed by three members of the Program Committee. Based on these reviews and on an open discussion, eight papers were accepted to be presented at the conference as full papers and to be included in this volume. We thank all authors for their contributions and hard work that made this event possible.

The conference program also involved short presentations of exploratory papers that are not included in these proceedings, and we wish to extend our thanks to the authors of the exploratory submissions.

We are indebted to the Steering Committee, Program Committee, and additional reviewers for their valuable help during the last months. We are very grateful for the support of the Local Organizing Committee. Finally, we acknowledge the excellent cooperation from the OASIcs team for their help in producing this volume in time for the conference.

27th IFIP WG 1.5 International Workshop on Cellular Automata and Discrete Complex Systems (AUTOMATA 2021).
Editors: Alonso Castillo-Ramirez, Pierre Guillon, and Kévin Perrot
OpenAccess Series in Informatics
OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

# Committees

## Program committee

- **Alonso Castillo-Ramirez** (Universidad de Guadalajara, Mexico) **co-chair**
- **Pierre Guillon** (CNRS & Université d'Aix-Marseille, France) **co-chair**

- Nathalie Aubrun (CNRS & Université Paris-Saclay, France)
- Jan Baetens (Ghent University, Belgium)
- Pedro Balbi de Oliveira (Universidade Presbiteriana Mackenzie, Brazil)
- Tullio Ceccherini-Silberstein (Università del Sannio, Italy)
- Alberto Dennunzio (Università di Milano Bicocca, Italy)
- Jeremias Epperlein (Universität Passau, Germany)
- Terry Farrelly (University of Queensland, Australia)
- Nazim Fatès (INRIA & Université de Lorraine, France)
- Enrico Formenti (Université de Nice-Côte-d'Azur, France)
- Anahí Gajardo (Universidad de Concepción, Chile)
- Janko Gravner (University of California Davis, US)
- Nataša Jonoska (University of South Florida, US)
- Jarkko Kari (University of Turku, Finland)
- Johan Kopra (University of Turku, Finland)
- Martin Kutrib (Universität Giessen, Germany)
- Kenichi Morita (University of Hiroshima, Japan)
- Kévin Perrot (Université d'Aix-Marseille, France)
- Trent Rogers (University of Arkansas, US)
- Véronique Terrier (Université de Caen-Normandie, France)
- Guillaume Theyssier (CNRS & Université d'Aix-Marseille, France)
- Thomas Worsch (Karlsruhe Institute of Technology, Germany)
- Reem Yassawi (Open University, UK)

## Additional reviewers

- Diego Maldonado
- Charalampos Zinoviadis

## Steering committee

- Pedro Balbi de Oliveira (Universidade Presbiteriana Mackenzie, Brazil)
- Jarkko Kari (University of Turku, Finland)
- Hector Zenil (Karolinska Institute, Sweden)
- Alonso Castillo-Ramirez (Universidad de Guadalajara, Mexico)
- Jan Baetens (Ghent University, Belgium)

## Local Organizing Committee

Pablo Arrighi, Amélia Durbec, Nathanaël Eon, Guilhem Gamard, Pierre Guillon, Etienne Moutot, Kévin Perrot, Antonio E. Porreca, Élisabeth Remy, Sylvain Sené, Guillaume Theyssier, from CNRS & Université d'Aix-Marseille.

27th IFIP WG 1.5 International Workshop on Cellular Automata and Discrete Complex Systems (AUTOMATA 2021).
Editors: Alonso Castillo-Ramirez, Pierre Guillon, and Kévin Perrot
OpenAccess Series in Informatics
OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

# Dynamical Properties of Disjunctive Boolean Networks

**Maximilien Gadouleau** ✉
Durham University, UK

## Abstract

A Boolean network is a mapping $f : \{0,1\}^n \to \{0,1\}^n$, which can be used to model networks of $n$ interacting entities, each having a local Boolean state that evolves over time according to a deterministic function of the current configuration of states. In this paper, we are interested in disjunctive networks, where each local function is simply the disjunction of a set of variables. As such, this network is somewhat homogeneous, though the number of variables may vary from entity to entity, thus yielding a generalised cellular automaton. The aim of this paper is to review some of the main results, derive some additional fundamental results, and highlight some open problems on the dynamics of disjunctive networks. We first review the different defining characteristics of disjunctive networks and several ways of representing them using graphs, Boolean matrices, or binary relations. We then focus on three dynamical properties of disjunctive networks: their image points, their periodic points, and their fixed points. For each class of points, we review how they can be characterised and study how many they could be. The paper finishes with different avenues for future work on the dynamics of disjunctive networks and how to generalise them.

**2012 ACM Subject Classification** Applied computing → Biological networks; Hardware → Quantum dots and cellular automata

**Keywords and phrases** Boolean networks, disjunction, conjunction, fixed points, rank

**Digital Object Identifier** 10.4230/OASIcs.AUTOMATA.2021.1

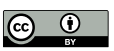
**Category** Invited Talk

## 1 Introduction

Consider a finite network of $n$ entities $1 \leq i \leq n$, where each has a Boolean state $x_i \in \{0,1\}$ that evolves over time according to a deterministic rule $f_i(x_1, \ldots, x_n) : \{0,1\}^n \to \{0,1\}$. The evolution of the configuration of states $x = (x_1, \ldots, x_n)$ is fully characterised by the global update function $f = (f_1, \ldots, f_n) : \{0,1\}^n \to \{0,1\}^n$. Such a function is called a Boolean network. Boolean networks are a versatile model and as such have been used to represent different networks, such as gene networks, neural networks, social networks, or network coding (see [16] and references therein for the applications of Boolean networks).

Cellular automata form a major model of discrete interactions. The network of entities (sometimes referred to as cells) is homogeneous, in the sense that the local update functions are all similar, and even though the number of cells is usually infinite, the interactions are only local. Cellular automata have attracted a huge amount of interest, both for their theoretical properties and their numerous applications (see [24, 27]).

Boolean networks and Cellular automata share some common typical characteristics, such as discrete space, discrete time, and finite state values for each entity (though some generalisations do not share all those characteristics). However, they differ in many aspects. Boolean networks are very general, with heterogeneous network topology, different local functions, finite number of entities, and various update schedules (synchronous or asynchronous). Cellular automata are much more restricted, with regular topology (usually a lattice $\mathbb{Z}^d$), the same local update function everywhere, an infinite number of cells, and typically parallel updates.



27th IFIP WG 1.5 International Workshop on Cellular Automata and Discrete Complex Systems (AUTOMATA 2021).
Editors: Alonso Castillo-Ramirez, Pierre Guillon, and Kévin Perrot; Article No. 1; pp. 1:1–1:15

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

There are many ways to possibly bridge the gap between these two models, by either relaxing some properties of CA or focusing on specific Boolean networks. In this paper, we are interested in a special class of Boolean networks where the local functions $f_i$ are "the same" everywhere, despite different cells interacting with more or less cells. In particular, we focus on disjunctive networks, where the local function at every entity is the disjunction of some variables:

$$f_i(x) = \bigvee_{j \in N(i)} x_j.$$

The set of variables $N(i)$ does depend on the entity and its size may vary.

Since disjunctions are such special Boolean functions, disjunctive networks can be characterised in different ways. We first review these characterisations in Section 2.2. Also, disjunctive networks can be represented using graphs, Boolean matrices, or binary relations; again, we review these representations in Section 2.3.

The dynamical properties of Boolean networks have been thoroughly studied, see [3, 6, 7, 17, 20, 22, 31] for example. Due to their different representations, disjunctive networks have attracted interest inside and outside of the Boolean network community. For instance, some early work on binary relations and Boolean matrices classifies convergent and idempotent disjunctive networks [10, 30, 35]. The main interests in the dynamical properties of disjunctive networks include the transient length [21], the characterisation of their cycle structure [21, 28], and in particular determining when they present no oscillations [1, 28]. It is worth noting that some results on disjunctive networks are included in works that consider related or more general classes of Boolean networks, e.g. [2, 3, 5, 7]. Even though in this paper we focus on the synchronous dynamics (i.e. parallel updates), the reader interested in the asynchronous dynamics of disjunctive networks is directed to [18].

In this paper, we are interested in the following three dynamical features of a disjunctive network. An image point of $f$ is a reachable state; a periodic point is a recurring state ; and a fixed point is a stationary state. We first give characterisations of their sets of image points, periodic points, and fixed points, respectively in Section 3.1. We then consider the number of image, periodic, and fixed points of disjunctive networks. We prove some results on the possible values these numbers can take in Section 3.2.

In this paper, we survey some of the main results on disjunctive networks, obtain some new results, and highlight some open problems in that area. The rest of this paper is organised as follows. Section 2 defines disjunctive networks, gives different characterisations of such networks, and illustrates how they can be represented. In Section 3, we study the dynamical properties of disjunctive networks when updated synchronously, with a focus on image points, periodic points, and fixed points. Finally, some possible avenues for generalisations and future work are given in Section 4.

# 2 Elementary properties

## 2.1 Definition

We denote the Boolean alphabet as $\mathbb{B} = \{0, 1\}$, which we endow with the natural order $0 < 1$. For any $x, y \in \mathbb{B}$, their **disjunction** is simply $x \vee y = \max\{x, y\}$. The disjunction operation satisfies the following properties (for all $x, y, z \in \mathbb{B}^n$):

- It is associative: $x \vee (y \vee z) = (x \vee y) \vee z$.
- It is commutative: $x \vee y = y \vee x$.
- It has an identity element, namely 0: $x \vee 0 = 0 \vee x = x$.

In view of these properties, we can generalise disjunction to an arbitrary number of Boolean variables. Let $S$ be a finite set and consider a configuration $x = (x_s : s \in S) \in \mathbb{B}^S$, then

$$\bigvee_{s\in S} x_s = \begin{cases} 1 & \text{if } \exists t \in S : x_t = 1 \\ 0 & \text{otherwise.} \end{cases}$$

In particular, if $S = \emptyset$, then $\bigvee_{s\in S} x_s = 0$; if $S = \{s\}$, then $\bigvee_{s\in S} x_s = x_s$.

The **conjunction** of $x, y \in \mathbb{B}$ is $x \wedge y = \min\{x, y\}$. Since conjunction and disjunction are dual, i.e. they are equivalent up to re-ordering 0 and 1, all the results we shall state about disjunction can be translated to apply to conjunction as well. In particular, we can define the conjunction of an arbitrary set of variables. For our purposes, it will be useful to group disjunction and conjunction under one common name. As such, we say that an operation is a **junction** if it is either a disjunction or a conjunction.

A point $x \in \mathbb{B}^n$ is called a **configuration**, which we shall denote as $x = (x_1, \dots, x_n)$ where $x_i \in \mathbb{B}$ for all $i \in [n] := \{1, \dots, n\}$. We introduce some further notation for configurations: we write $x = (x_i, x_{-i})$ where $x_{-i} = (x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n) \in \mathbb{B}^{n-1}$, and we define the unit configuration $e^j$ to satisfy $e^j_j = 1$, $e^j_{-j} = 0_{-j}$. The Hamming distance between $x, y \in \mathbb{B}^n$ is the number of coordinates they disagree: $d_{\mathrm{H}}(x, y) = |\{i \in [n] : x_i \neq y_i\}|$; the Hamming weight of $x$ is the number of ones in $x$: $w_{\mathrm{H}}(x) = |\{i : x_i = 1\}|$.

A **Boolean network** of dimension $n$ is a mapping $f : \mathbb{B}^n \to \mathbb{B}^n$. We also split $f$ as $f = (f_1, \dots, f_n)$ where $f_i : \mathbb{B}^n \to \mathbb{B}$ is a Boolean function representing the update of the state $x_i$ of the $i$-th entity of the network. A Boolean network $f$ is **disjunctive** if $f_i$ is a disjunction for all $i \in [n]$.

A (directed) graph $D = (V, E)$ is a pair where $V$ is the set of vertices and $E \subseteq V^2$ is the set of arcs of $D$ [8]. In this paper, we only consider finite graphs and we shall usually identify isomorphic graphs. For any vertex $i \in V$, its in-neighbourhood in $D$ is $N^{\mathrm{in}}(i) = \{j \in V : (j, i) \in E\}$ and its in-degree is the size of its in-neighbourhood; the out-neighbourhood an out-degree are defined similarly. A vertex is a source (sink, respectively) if its in-neighbourhood (out-neighbourhood, respectively) is empty.

The **interaction graph** of $f$, denoted as $\mathbb{D}(f)$, represents the influences of entities on one another. Formally, $\mathbb{D}(f) = (V, E)$, where $V = [n]$ and $(i, j) \in E$ if and only if $f_j$ depends essentially on $x_i$, i.e. there exists $a_{-i}$ such that

$$f_j(0, a_{-i}) \neq f_j(1, a_{-i}).$$

If $D$ is the interaction graph of $f$, we then say that $f$ is a Boolean network on $D$. Clearly, for every graph $D$ there is a unique disjunctive network on $D$.

## 2.2 Characterisations

We can extend the order $0 < 1$ in $\mathbb{B}$ to configurations $x, y \in \mathbb{B}^n$ componentwise: we write $x \leq y$ if and only if $x_i \leq y_i$ for all $i \in [n]$. We can also extend the disjunction notation to configurations by applying it componentwise: $x \vee y = z$ with $z_i = x_i \vee y_i$ for all $i \in [n]$. We then have

$$x \leq y \iff x \vee y = y. \tag{1}$$

A Boolean function $\phi : \mathbb{B}^n \to \mathbb{B}$ is **monotone** if $x \leq y$ implies $\phi(x) \leq \phi(y)$. It is easily checked that any junction is monotone. We say a Boolean network $f$ is **monotone** if $x \leq y$ implies $f(x) \leq f(y)$; clearly $f$ is monotone if and only if $f_i$ is monotone for all $i$.

▶ **Lemma 1.** *[25, Theorem 11.1] A Boolean network $f$ is monotone if and only if for all $x, y \in \mathbb{B}^n$,*

$$f(x \vee y) \geq f(x) \vee f(y). \tag{2}$$

**Proof.** Suppose $f : \mathbb{B}^n \to \mathbb{B}^n$ is monotone, then $f(x \vee y) \geq f(x)$ and $f(x \vee y) \geq f(y)$, hence $f(x \vee y) \geq f(x) \vee f(y)$. Conversely, if $f(x \vee y) \geq f(x)$ for all $x, y$, then for any $a \leq b$ we have $f(b) = f(a \vee b) \geq f(a)$, hence $f$ is monotone. ◀

Combining (1) and (2), we obtain a characterisation of monotone networks based on an equation.

▶ **Corollary 2.** *[25, Example 11.5] A Boolean network $f$ is monotone if and only if for all $x, y \in \mathbb{B}^n$,*

$$f(x \vee y) = f(x) \vee f(x \vee y).$$

The first, and arguably canonical, characterisation of disjunctive networks is that they are the endomorphisms of the disjunction on $\mathbb{B}^n$. As such, disjunctive networks are those that reach equality in (2) and that fix the all-zero configuration. The latter property is a technical detail, which comes the fact that the constant Boolean function $\phi(x) = 1$ is not a disjunction and yet also satisfies $\phi(x \vee y) = \phi(x) \vee \phi(y)$.

▶ **Theorem 3.** *A Boolean network $f$ is disjunctive if and only if $f(0, \ldots, 0) = (0, \ldots, 0)$ and for all $x, y \in \mathbb{B}^n$,*

$$f(x \vee y) = f(x) \vee f(y).$$

**Proof.** The forward implication is straightforward; we focus on the reverse implication. Suppose $f$ fixes $(0, \ldots, 0)$ and $f(x \vee y) = f(x) \vee f(y)$ for all $x, y \in \mathbb{B}^n$. First, by Lemma 1, $f$ is monotone. If $f$ is not disjunctive, then $f_i$ is not a disjunction for some $i$, and hence by monotonicity there exists $j \in N^{\text{in}}(i)$ such that $f_i(e^j) = 0$. There exists $a_{-j}$ such that $f_i(0, a_{-j}) = 0$ and $f_i(1, a_{-j}) = 1$. We obtain

$$\begin{aligned} f_i((0, a_{-j}) \vee e^j) &= f_i(1, a_{-j}) = 1, \\ f_i(0, a_{-j}) \vee f_i(e^j) &= 0 \vee 0 = 0, \end{aligned}$$

which is the desired contradiction. ◀

The second characterisation is that disjunctive networks are precisely the submodular monotone networks. A Boolean network is **submodular** if for all $x, y \in \mathbb{B}^n$,

$$f(x \vee y) \vee f(x \wedge y) \leq f(x) \vee f(y).$$

Submodular Boolean functions form an important class of Horn functions; the interested reader is directed to [9, Section 6.9.1] and references therein. Theorem 3 then immediately yields a second characterisation of disjunctive networks, implicit from Theorems 11.1 and 11.4 in [25].

▶ **Theorem 4.** *A Boolean network is disjunctive if and only if it is monotone and submodular and it fixes the all-zero configuration.*

The third characterisation of disjunctive networks is that of maximal locally idempotent networks. A Boolean function $\phi : \mathbb{B}^n \to \mathbb{B}$ is **idempotent** if $\phi(0, \dots, 0) = 0$ and $\phi(1, \dots, 1) = 1$ [32]. We then say a Boolean network $f$ is **locally idempotent** if $f_i$ is idempotent for all $i$; equivalently $f$ is locally idempotent if it fixes the all-zero and the all-one configurations. Since an idempotent Boolean function is not constant, the interaction graph of a locally idempotent network has no sources. Conversely, any monotone network on a graph with no sources is locally idempotent. For two Boolean networks $f, g : \mathbb{B}^n \to \mathbb{B}^n$, we naturally write $f \leq g$ if $f(x) \leq g(x)$ for all $x \in \mathbb{B}^n$.

▶ **Theorem 5.** *Let $f$ be a locally idempotent network with interaction graph $D$ and let $f^\wedge$ and $f^\vee$ be the conjunctive and disjunctive networks on $D$, respectively. Then*

$$f^\wedge \leq f \leq f^\vee.$$

**Proof.** For all $x \in \mathbb{B}^n$ and all $i \in [n]$,

$$f_i(x) = 0 \implies x_{N^{\mathrm{in}}(i)} \neq (1, \dots, 1) \implies f_i^\wedge(x) = 0,$$
$$f_i(x) = 1 \implies x_{N^{\mathrm{in}}(i)} \neq (0, \dots, 0) \implies f_i^\vee(x) = 1.$$

This yields $f_i^\wedge(x) \leq f_i(x) \leq f_i^\vee(x)$. ◀

The fourth characterisation of disjunctive networks is to be "closest" to constant networks– technically this does not fully characterise disjunctive networks, as any network where the local function is a junction satisfies this property. The **distance** between two Boolean networks $f, g : \mathbb{B}^n \to \mathbb{B}^n$ is

$$d(f, g) := \sum_{x \in \mathbb{B}^n} d_{\mathrm{H}}(f(x), g(x)).$$

A Boolean network $g$ is **constant** if there exists $c \in \mathbb{B}^n$ for which $g(x) = c$ for all $x \in \mathbb{B}^n$; we denote the set of constant networks as $C$. The **distance to constant networks** of $f$ is then defined as

$$\begin{aligned} d_C(f) &:= \min\{d(f, g) : g \in C\} \\ &= \min_{c \in \mathbb{B}^n} \sum_{x \in \mathbb{B}^n} d_{\mathrm{H}}(f(x), c). \end{aligned}$$

Then $d_C(f)$ ranges from 0 (whenever $f$ is constant) to $n2^{n-1}$ (whenever $f$ is a permutation, amongst others).

▶ **Theorem 6.** *Let $f$ be a Boolean network with interaction graph $D$ and let $f^\wedge$ and $f^\vee$ be the conjunctive and disjunctive networks on $D$, respectively. Then*

$$d_C(f) \geq d_C(f^\vee) = d_C(f^\wedge).$$

**Proof.** For any $a, b \in \mathbb{B}$, we denote $a \oplus b = a + b \mod 2$, i.e. $a \oplus b = 1$ if and only if $a = \neg b$. We have

$$\begin{aligned} d_C(f) &= \min_{c \in \mathbb{B}^n} \sum_{x \in \mathbb{B}^n} \sum_{i \in [n]} f_i(x) \oplus c_i \\ &= \min_{c \in \mathbb{B}^n} \sum_{i \in [n]} |f_i^{-1}(\neg c_i)|. \end{aligned}$$

For any $i$, let $d_i$ denote the in-degree of $i$ in $D$; let $T$ denote the set of non-sources of $D$. We have

$$\begin{aligned} d_C(f) &= \min_{c\in\mathbb{B}^n} \sum_{i\in[n]} |f_i^{-1}(\neg c_i)| \\ &= \sum_{i\in[n]} \min\{|f_i^{-1}(0)|, |f_i^{-1}(1)|\} \\ &= \sum_{i\in T} \min\{|f_i^{-1}(0)|, |f_i^{-1}(1)|\} \\ &\geq \sum_{i\in T} 2^{n-d_i}. \end{aligned}$$

It is clear that the last inequality is reached for the disjunctive (or conjunctive) network on $D$. ◀

## 2.3 Representations

Let $f$ be the disjunctive network on $D = (V = [n], E)$. We give below four representations of $f$.

**Boolean linear mapping** A graph can be represented by its adjacency matrix $A_D$ where $a_{i,j} = 1$ if and only if $(i, j) \in E$. The product of two Boolean matrices is $AB = C$ where

$$c_{ij} = \bigvee_{k=1}^{n} a_{ik} \wedge b_{kj}.$$

Boolean matrix theory has been widely studied from an algebraic and combinatorial point of view, and has found applications in different areas of computer science; the interested reader is directed to the authoritative book [30]. A **Boolean linear mapping** is any $g : \mathbb{B}^n \to \mathbb{B}^n$ of the form $g(x) = xA$ for some Boolean matrix $A$. In our case, the disjunctive network on $D$ satisfies $f(x) = xA_D$. Since any Boolean matrix is the adjacency matrix of some graph, disjunctive networks are exactly the Boolean linear mappings. We remark that Boolean linear mappings are different from their finite field counterparts, which are usually simpler to analyse.

**Out-neighbourhood function** Identifying $x \in \mathbb{B}^n$ with its support $X = \mathrm{supp}(x) = \{i \in [n] : x_i = 1\}$, we can identify $f$ with the mapping on the power set of $[n]$ defined by

$$f(X) = N^{\mathrm{out}}(X).$$

**Binary relation** A **binary relation** $R$ on $[n]$ is a subset of $[n] \times [n]$. Binary relations are in one-to-one correspondence with Boolean matrices; as such, the semigroup of binary relations has been widely studied [26]. Clearly graphs are also in one-to-one correspondence with binary relations (the edge set $E$ of $D$ is a binary relation). We can then represent $f$ as $f(X) = \{y \in [n] : \exists s \in X, (s, y) \in E\}$.

**Token sliding** We can rewrite the representation above as

$$f_i(X) = \bigcup_{j\in N^{\mathrm{in}}(i)} X_j.$$

This can be interpreted as follows. Suppose there are $n$ tokens $T = \{t_1, \ldots, t_n\}$ on the $n$ vertices of $D$. Let $X_i \subseteq T$ denote the collection of tokens that $i$ holds at a given time. At each time step every vertex $i$ broadcasts all its tokens to all possible destinations. Then vertex $j$ obtains all tokens from its in-neighbours, and hence $X_j$ becomes $\bigcup_{j\in N^{\mathrm{in}}(i)} X_j$.

# 3 Dynamical properties

Let $f$ be a Boolean network. We consider three types of points for $f$:

- An **image point** of $f$ is $x \in \mathbb{B}^n$ such that $f(y) = x$ for some $y \in \mathbb{B}^n$. In the transformation semigroup literature, the number of image points is usually called "rank". However, different ranks for the matrix $A_D$ have been proposed [30]. Therefore, in order to emphasize that we are counting the number of image points, we shall use refer to the number of image points of $f$ as its **image rank**.
- A **periodic point** of $f$ is $x \in \mathbb{B}^n$ such that $f^k(x) = x$ for some $k \geq 1$. The number of periodic points of $f$ is called the **periodic rank** of $f$. Clearly, the periodic rank of $f$ is equal to the image rank of $f^{2^n}$, and is less than or equal to the image rank of $f^p$ for any $p \geq 1$.
- A **fixed point** of $f$ is $x \in \mathbb{B}^n$ such that $f(x) = x$. The number of fixed points of $f$ is called the **fixed rank** of $f$. Unlike the image and periodic ranks, the fixed rank of a Boolean network can be equal to zero.

## 3.1 Image, periodic and fixed points

In this subsection, we review some of the key results describing the sets of image points, periodic points, and fixed points of disjunctive networks. It will be convenient to identify a Boolean configuration $x \in \mathbb{B}^n$ with its support $X = \{i \in [n] : x_i = 1\} \subseteq [n]$ and to view the disjunctive network on $D$ as $f : X \mapsto N^{\mathrm{out}}(X)$.

### 3.1.1 Image points

Given a disjunctive network $f$ and a subset $X$, it is easy to verify whether $X$ is an image point of $f$. By definition, $X$ belongs to the image of $f$ if it is the out-neighbourhood of some set of vertices: $X = N^{\mathrm{out}}(Y)$ for some $Y \subseteq [n]$. The key property is that there is a unique maximal preimage $Y^*$ of $X$, so we only need to compute $f(Y^*)$ and check whether it matches with $X$.

▶ **Proposition 7.** *For any $X \subseteq [n]$, let*

$$Y^* := [n] \setminus \left(N^{\mathrm{in}}([n] \setminus X)\right).$$

*Then $X$ is an image point of $f$ if and only if $X = N^{\mathrm{out}}(Y^*)$. If that is the case, then*

$$Y^* = \bigcup_{Y \in f^{-1}(X)} Y.$$

**Proof.** Suppose $X = f(Y)$ for some $Y$. Firstly, $Y \subseteq Y^*$, for if $j \in Y \cap N^{\mathrm{in}}(i)$ for some $i \notin X$, then $i \in N^{\mathrm{out}}(Y)$. Therefore, $X \subseteq f(Y^*)$. Conversely, $N^{\mathrm{out}}(Y^*) \subseteq X$. Indeed, for any $j \notin X$, $N^{\mathrm{in}}(j) \cap Y^* = \emptyset$ and hence $j \notin N^{\mathrm{out}}(Y^*)$. Combining, we obtain $X = f(Y^*)$.

The second statement follows from the fact that $f(Y) = X$ implies $Y \subseteq Y^*$. ◀

Determining the image rank of a function of the form $f(x) = xM$, where $M$ is over a finite field $\mathrm{GF}(q)$, is simple: it is given by $q^{\mathrm{rank}(M)}$, where the rank can be computed in polynomial time. On the other hand, it is not even clear whether computing the image rank of a disjunctive network can be done in polynomial time.

▶ **Problem 1.** *What is the complexity of the following problem: given a graph $D$, what is the image rank of the disjunctive network on $D$?*

### 3.1.2 Periodic points

The set of periodic points of the disjunctive network $f$ on $D$ have been studied in [21, 28, 1]. In order to keep the notation simple and to give the intuition behind the main results, we focus on the case where $D$ is **strong** (a.k.a. strongly connected), i.e. there is a path from $i$ to $j$ for any two vertices $i$ and $j$. For results about general graphs, see [28].

The **loop number** $l(D)$ of $D$ is the greatest common denominator of all the cycle lengths in $D$. If $l(D) = 1$ and $D$ is strong, we say that $D$ is **primitive** (graphs with loop number one are sometimes called aperiodic). It is well known that if $l(D) = 2$, then the graph is bipartite: we can partition its vertex set in two parts such that all the arcs go between the parts. This is generalised as follows.

▶ **Lemma 8.** *[8, Theorem 17.8.1] If a strong digraph $D = (V, E)$ has loop number $l(D) = l \geq 2$, then $V$ can be partitioned into $V_0, \ldots, V_{l-1}$ such that $N^{\text{out}}(V_i) = V_{i+1}$ (indices computed modulo $l$).*

Say a subset of vertices $X$ is **$D$-partite** if

$$X = V_{i_1} \cup \cdots \cup V_{i_a}$$

for some $i_1, \ldots, i_a \in [l]$. Clearly, $X$ is a periodic point, since $f^l(X) = X$. The key result is that a subset is a periodic point if and only if it is $D$-partite. Indeed, let $Y$ be non-$D$-partite and let $s \in Y \cap V_i$ and $t \in V_i \setminus Y$. There is a path from $s$ to $t$; that path must have length equal to $kl$ for some $k$, hence $t \in f^{kl}(Y)$. Generalising this idea, we obtain that $f^q(Y)$ will eventually be $D$-partite for $q$ large enough.

▶ **Theorem 9.** *[21] Let $D$ be a strong graph, then a subset $X$ of vertices is a periodic point of the disjunctive network $f$ on $D$ if and only if $X$ is $D$-partite.*

The **period** of a periodic point $X$ is the smallest $p \geq 1$ such that $f^p(X) = X$. We have seen that $f^{l(D)}(X) = X$, which immediately yields:

▶ **Corollary 10.** *[21, 28] The period of a periodic point of $f$ divides $l(D)$. Conversely, for any $p \mid l(D)$, there is a periodic point of period $p$.*

In particular, if $D$ is primitive, then the only periodic points are $\emptyset$ and $[n]$, which are fixed points. More results can be obtained, for instance the time it takes to reach a periodic points can be upper bounded [21] and the number of periodic points of a certain period can be determined [28].

### 3.1.3 Fixed points

We give a classification of the set of fixed points of a disjunctive network. In order to keep it simple, we focus on **nontrivial** graphs, where every vertex belongs to a cycle.

The axioms of topology simplify greatly for finite spaces: a collection $T$ of subsets of $[n]$ is a **topology** on $[n]$ if and only if $\emptyset, [n] \in T$ and

$$X, Y \in T \implies X \cup Y, X \cap Y \in T.$$

Let $D = ([n], E)$ be a nontrivial graph and for any $i, j \in [n]$ denote $i \leq j$ if there is a path from $i$ to $j$. We note that $\leq$ is only a preorder relation, and that reflexivity is guaranteed by the fact that $D$ is nontrivial. For any $S \subseteq [n]$, let the **up-set** of $S$ to be the set of vertices reachable from $S$:

$$S^{\uparrow} := \{j : \exists i \in S, i \leq j\}.$$

It is easily checked that the collection of up-sets forms a topology. We denote this collection as

$$\mathbb{T}(D) := \{S^{\uparrow} : S \subseteq [n]\}.$$

▶ **Theorem 11.** *The set of fixed points of the disjunctive network on the nontrivial graph $D$ is $\mathbb{T}(D)$.*

**Proof.** The subset $X$ is a fixed point of $f$ if and only if $X = N^{\text{out}}(X)$, that is $X = S^{\uparrow}$ for some $S$. ◀

In fact, any finite topology arises from a nontrivial graph; this well known result is usually given in terms of preorders (see [11]).

▶ **Lemma 12.** *[11, Theorem 3.9.1] Let $T$ be a topology on $[n]$, then $T = \mathbb{T}(D)$ for some nontrivial graph $D$ on $[n]$.*

The Knaster-Tarski theorem asserts that the set of fixed points of a monotone network forms a lattice. Conversely, for any lattice of configurations of $\mathbb{B}^n$, it is easy to construct a monotone network whose set of fixed points is exactly that lattice. Therefore, a subset of $\mathbb{B}^n$ is the set of fixed points of a monotone network if and only if it forms a lattice. We obtain a similar characterisation for the sets of fixed points of disjunctive networks.

▶ **Corollary 13.** *$T$ is the set of fixed points of a disjunctive network on a nontrivial graph if and only if it is a topology.*

## 3.2 Values of the image, periodic and fixed ranks

In this subsection, we consider the values that can be taken by the different ranks of a disjunctive network. Foremost, a Boolean network is **idempotent** if $f^2 = f$, i.e. $f(f(x)) = f(x)$ for all $x \in \mathbb{B}^n$. It is clear that a Boolean network is idempotent if and only if its fixed rank, periodic rank and image rank are all equal. Idempotent disjunctive networks were characterised, under the guise of binary relations, by Rosenblatt [35] (see [10]).

The values the image/periodic/fixed rank can take for a monotone network are easy to characterise. Note that the Knaster-Tarski theorem implies that any monotone network has at least one fixed point.

▶ **Proposition 14.** *For any $n$ and any $1 \leq k \leq 2^n$, there exists an idempotent monotone network of dimension $n$ with exactly $k$ image points.*

**Proof.** Sort the configurations of $\mathbb{B}^n$ in non-decreasing order of Hamming weight, so that $x^0 = (0, \dots, 0), \dots, x^{2^n-1} = (1, \dots, 1)$ and $x^i \leq x^j$ implies $i \leq j$. Let $f$ be defined as

$$f(x^a) = \begin{cases} x^a & \text{if } 0 \leq a \leq k-2 \\ x^{2^n-1} & \text{if } k-1 \leq a \leq 2^{n-1}. \end{cases}$$

Then $f$ is idempotent, monotone and has image/periodic/fixed rank $k$. ◀

On the other hand, the situation for disjunctive networks is more complex. We shall obtain some results on the values the image/periodic/fixed rank of a disjunctive network can take but we will remain far from classifying them.

▶ **Problem 2.** *What is the complexity of the following problem: given $k$ and $n$, is there a disjunctive network of dimension $n$ with image rank $k$?*

First, we consider the maximum value of the image/periodic/fixed ranks. Clearly, it is given by $2^n$, but we can even classify the disjunctive networks with image/periodic rank $2^n$. (There is only one Boolean network with fixed rank $2^n$, namely the identity, which happens to be disjunctive.) In fact, the only bijective monotone networks are **permutations of variables**. This classification is folklore, and is related to the classification of isometries of the hypercube. The symmetric group on $[n]$ acts naturally on configurations in $\mathbb{B}^n$, whereby $\pi(x) = (x_{\pi(1)}, \dots, x_{\pi(n)})$ for any permutation $\pi$ of $[n]$. Any such Boolean network is called a permutation of variables. Note that permutations of variables are disjunctive networks: they correspond to interaction graphs that are disjoint unions of cycles.

▶ **Theorem 15.** *A monotone Boolean network is bijective if and only if it is a permutation of variables.*

**Proof.** Let $f$ be a bijective monotone Boolean network. We first prove that $f$ preserves the Hamming weight. Any $x \in \mathbb{B}^n$ of Hamming weight $k$ belongs to a maximal chain

$$x^0 = (0, \dots, 0) < x^1 < \cdots < x^k = x < \cdots < x^n = (1, \dots, 1),$$

where $w_{\mathrm{H}}(x^i) = i$ for all $0 \leq i \leq n$. By monotonicity and injectivity, we have

$$f(x^0) < f(x^1) < \cdots < f(x^n),$$

whence $w_{\mathrm{H}}(f(x^i)) = i$ for all $0 \leq i \leq n$, and in particular $w_{\mathrm{H}}(f(x)) = x$.

We now prove that $f(x) = \pi(x)$ for some permutation $\pi$, by induction on $k = w_{\mathrm{H}}(x)$. The base cases $k = 0$ and $k = 1$ are clear, so let us assume $k \geq 2$ and that it holds for up to $k - 1$. Suppose $x_i = x_j = 1$, and denote $x^{-i} = (0, x_{-i})$ and $x^{-j} = (0, x_{-j})$. We have

$$f(x) \geq f(x^{-i}) \vee f(x^{-j}) = \pi(x^{-i}) \vee \pi(x^{-j}) = \pi(x).$$

Since $f(x)$ and $\pi(x)$ both have Hamming weight $k$, we have equality: $f(x) = \pi(x)$. ◀

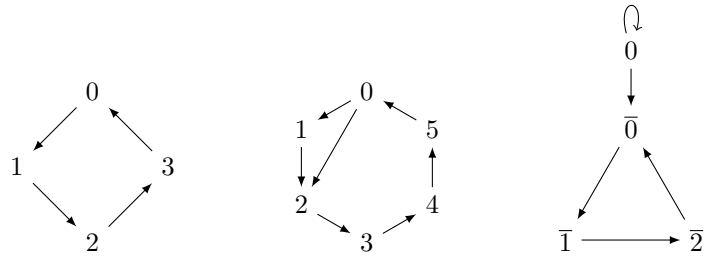
We now move on to singular networks, i.e. those that are not bijective. Even though there are monotone Boolean networks of image rank $k$ for all $1 \leq k \leq 2^n$, and in particular for $k = 2^n - 1$, the image rank of singular disjunctive networks is upper bounded by $3/4 \cdot 2^n$. We fully classify the graphs that reach the upper bound in Theorem 16 below. The classification is based on the following three families of connected graphs:

1. $C_n$ ($n \geq 1$) is the cycle on $n$ vertices, with vertex set $V = \mathbb{Z}_n$ and arcs $E = \{(i, i+1 \bmod n) : i \in \mathbb{Z}_n\}$ for $n \geq 2$ and $E = \{(0,0)\}$ for $n = 1$.
2. $A_{p,q}$ ($0 \leq q \leq p-1$) is the chorded cycle: the cycle $C_p$, augmented by the chord $\{(0, q)\}$.
3. $B_{s,t}$ ($s, t \geq 1$) is the link of cycles: formed of two cycles $C_s$ and $C_t$, with a single arc from $C_s$ to $C_t$. We denote the vertices of $C_s$ as $0, \dots, s-1$ and those of $C_t$ as $\overline{0}, \dots, \overline{t-1}$.

Examples of these graphs ($C_4$, $A_{6,2}$ and $B_{1,3}$) are displayed on Figure 1 . We then say that a graph is **near-cyclic** if one of its connected components is a chorded cycle or a link of cycles, and all other connected components are cycles. More formally, $D$ is near-cyclic if it is of the form $D = A_{p,q} \cup C_{n_1} \cup \cdots \cup C_{n_c}$ or $D = B_{s,t} \cup C_{n_1} \cup \cdots \cup C_{n_c}$ for some choice of parameters.

▶ **Theorem 16.** *For $n \geq 2$, the maximum image rank of a singular disjunctive network of dimension $n$ is $3/4 \cdot n$, and it is reached if and only if its interaction graph is near-cyclic.*

**Proof.** We first prove that the image rank is upper bounded by $3/4 \cdot 2^n$. Let $f$ be a singular disjunctive network and $D$ be its interaction graph.




**(a)** $C_4$ **(b)** $A_{6,2}$ **(c)** $B_{1,3}$

**Figure 1** Graphs used in Theorem 16.

If all the vertices of $D$ have in-degree one, then it must have a sink $k$ (otherwise, $f$ would be a permutation of variables). But then, any two configurations only differing in position $k$ have the same image under $f$, and hence $|f(\mathbb{B}^n)| \leq \frac{1}{2}2^n$.

If $D$ has a vertex $i$ of in-degree $d \geq 2$, then $|f_i^{-1}(0)| = 2^{n-d}$, and hence

$$\begin{aligned}|f(\mathbb{B}^n)| &\leq |\{y \in \mathbb{B}^n : y_i = 1\}| + |f_i^{-1}(0)| \\ &\leq 2^{n-1} + 2^{n-d} \\ &\leq 3/4 \cdot 2^n.\end{aligned}$$

If $D$ has a source $i$, then $|f_i^{-1}(1)| = 0$, and we similarly obtain $|f(\mathbb{B}^n)| \leq 1/2 \cdot 2^n$.

We now characterise the graphs that reach the upper bound. By the above, there is a vertex of in-degree 2. We first prove that it is unique. Suppose for the sake of contradiction that both $i$ and $j$ have in-degree 2. We do a case analysis based on $|N^{\text{in}}(i,j)|$. For the sake of simplicity, we write $\{ab = 01, 10\}$ as a shorthand for $\{y \in \mathbb{B}^n : y_a y_b \in \{01, 10\}\}$ and we extend this shorthand notation to arbitrary sets of coordinates and values.

- $|N^{\text{in}}(i,j)| = 2$. Then $f_i = f_j$ and hence

$$|f(\mathbb{B}^n)| \leq |\{ij = 00, 11\}| = \frac{1}{2}2^n.$$

- $|N^{\text{in}}(i,j)| = 3$. Say $N^{\text{in}}(i) = \{a,b\}$ and $N^{\text{in}}(j) = \{a,c\}$. Then

$$\begin{aligned}|f(\mathbb{B}^n)| &= |f(\{abc = 000, 001, 010\})| + |f(\{abc = 011, 100, 101, 110, 111\})| \\ &\leq |\{abc = 000, 001, 010\}| + |\{ij = 11\}| \\ &= \frac{5}{8}2^n.\end{aligned}$$

- $|N^{\text{in}}(i,j)| = 4$. Say $N^{\text{in}}(i) = \{a,b\}$ and $N^{\text{in}}(j) = \{c,d\}$. Then

$$\begin{aligned}|f(\mathbb{B}^n)| &= |f(\{abcd = 0000, 0001, 0010, 0011, 0100, 1000, 1100\})| \\ &\quad + |f(\{abcd = 0101, 0110, 0111, 1001, 1010, 1011, 1101, 1110, 1111\})| \\ &\leq |\{abcd = 0000, 0001, 0010, 0011, 0100, 1000, 1100\}| + |\{ij = 11\}| \\ &= \frac{11}{16}2^n.\end{aligned}$$

We can now prove the result for reflexive graphs. A graph is **reflexive** if $(i,i) \in E$ for all $i \in V$. The only reflexive graph with a unique vertex of in-degree 2 is $G_n = B_{1,1} \cup C_1 \cup \cdots \cup C_1$, displayed on Figure 2; it is easy to check that the disjunctive network on $G_n$ indeed has image rank $3/4 \cdot 2^n$.

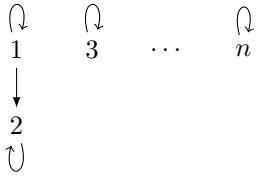



**Figure 2** The graph $G_n$.

Let $G = (V, E)$ be a graph and $\pi$ be a permutation of $V$. We define $D = G\pi$ the graph with vertex set $V$ and arc set $\{(i, \pi(j)) : (i, j) \in E\}$. In general, if a graph $D$ admits a Boolean network with image rank greater than $1/2 \cdot 2^n$, then it must be coverable by cycles [14], i.e. there exists a permutation $\pi$ of $[n]$ such that $D = G\pi$, where $G$ is reflexive. Thus, if $D$ is the interaction graph of a disjunctive network of image rank $3/4 \cdot 2^n$, then it is of the form $D = G_n\pi$ for some permutation $\pi$. If $\pi(1)$ and $\pi(2)$ belong to the same cycle of $\pi$, we obtain $D = A_{p,q} \cup C_{n_1} \cup \cdots \cup C_{n_c}$; otherwise we obtain $D = B_{s,t} \cup C_{n_1} \cup \cdots \cup C_{n_c}$. ◀

We also obtain the corresponding result for the periodic rank.

▶ **Corollary 17.** *For $n \geq 2$, the maximum periodic rank of a singular disjunctive network of dimension $n$ is $3/4 \cdot 2^n$, and reached if and only if the interaction graph is of the form $B_{1,1} \cup C_{n_1} \cup \cdots \cup C_{n_c}$.*

**Proof.** Let $f$ be a disjunctive network on the near-cyclic graph $D$, and let $D^2$ denote the interaction graph of $f^2$. If $D$ has an $A_{p,q}$ component, then $D^2$ has two vertices of in-degree at least two, namely $q$ and $q + 1$. A similar argument holds for a $B_{s,t}$ component with $t \geq 2$. For a $B_{s,1}$ component with $s \geq 2$, the vertex $\overline{0}$ has in-degree 3 in $D^2$. Therefore, in any case, the image rank of $f^2$ is less than $3/4 \cdot 2^n$, and so is the periodic rank of $f$. ◀

The corresponding result for the fixed rank immediately follows.

▶ **Corollary 18.** *For $n \geq 2$, the maximum fixed rank of a singular disjunctive network of dimension $n$ is $3/4 \cdot 2^n$, and reached if and only if the interaction graph is $G_n$.*

**Proof.** If a disjunctive network is singular, then its fixed rank is at most $3/4 \cdot 2^n$ by Corollary 17. Therefore, we can restrict ourselves to disjunctive networks on near-cyclic graphs of the form $B_{1,1} \cup C_{n_1} \cup \cdots \cup C_{n_c}$. It is clear that if any cycle $C_{n_i}$ has $n_i \geq 2$, then the disjunctive network has periodic points that are not fixed points; conversely the disjunctive network on $G_n$ is idempotent. ◀

We next consider the opposite problem: what is the smallest "missing value" of the image/periodic/fixed rank? Obviously, we consider a nonzero fixed rank.

▶ **Problem 3.** *Given $n$, what is the minimum $k \geq 1$ such that there is no disjunctive network of dimension $n$ and image/periodic/fixed rank $k$?*

We give a lower bound on that quantity below, for all three ranks.

▶ **Theorem 19.** *For any $n$, there exists an idempotent disjunctive network on $n$ vertices with image, periodic, and fixed ranks $r$ for all $1 \leq r \leq p - 1$, where $p$ is the smallest prime number greater than $n + 1$.*

**Proof.** The reflexive transitive tournament $T_a$ on $a$ vertices has arcs $ij$ for all $i \leq j$. Let $f$ be the disjunctive network on $T_a$, then it is easily seen that $f$ is idempotent and that its image is $\{\bigvee_{i=j}^{a} e^i : 1 \leq j \leq a+1\}$.

We can now prove the result; it is clear for $n \leq 2$ so we suppose $n \geq 3$. First, suppose $r \leq n+1$. We denote the empty graph on $c$ vertices as $E_c$. Then the disjunctive network on $T_{r-1} \cup E_{n-r+1}$ is idempotent and has image rank $r$. Second, suppose $n+2 \leq r \leq p-1$. Then by Bertrand's postulate, $p \leq 2n-1$ and hence $r \leq 2n-2$. Since $r$ is composite, we have $r = ab$ for $a+b \leq 2+r/2 \leq n+1$. Thus the disjunctive network on the graph $T_{a-1} \cup T_{b-1} \cup E_{n-a-b+2}$ is idempotent and has image rank $r$. ◀

# 4 Outlook

## 4.1 Disjunctive networks compared to other networks

It would be interesting to compare the disjunctive network on $D$ with the other Boolean networks on $D$. Firstly, we want to investigate when the disjunctive network has as many image/periodic/fixed points as possible. There are three main upper bounds on the image/periodic/fixed rank of a Boolean network with interaction graph $D$, that depend on three graph parameters reviewed in [15]. The image rank of a Boolean network $f$ on $D$ is at most $2^{\alpha_1(D)}$ [14], while its periodic rank is at most $2^{\alpha_n(D)}$ [14], and its fixed rank is at most $2^{\tau(D)}$ (the famous feedback bound [33, 4]). Those bounds are not always reached (e.g. the pentagon does not reach any). The graphs where the feedback bound is reached by a monotone network are classified in [7]. Similar classification results for disjunctive networks seem close at hand.

▶ **Problem 4.** *Classify the graphs $D$ such that:*
1. *the image rank of the disjunctive network on $D$ is $2^{\alpha_1(D)}$.*
2. *the periodic rank of the disjunctive network on $D$ is $2^{\alpha_n(D)}$.*
3. *the fixed rank of the disjunctive network on $D$ is $2^{\tau(D)}$.*

Secondly, we want to investigate when the disjunctive network minimises the image/periodic/fixed rank. Since disjunctive networks are the closest to being constant by Theorem 6, one might expect that they should minimise the image rank over all networks with a given interaction graph. This is true in the following extreme cases, where the minimum rank is equal to 1, 2, or $2^n$ [15]. However, this turns out not to be the case in general: [15, Theorem 5] gives a counter-example, where the minimum image rank is not achieved by the disjunctive network, but can be actually achieved by another monotone network. We therefore ask the following two questions.

▶ **Problem 5.** *Classify the graphs $D$ such that the disjunctive network on $D$ minimises the image rank over*
1. *all Boolean networks on $D$;*
2. *all monotone networks on $D$.*

For periodic points, we do not know which graphs $D$ admit a Boolean network with a single periodic point (the so-called nilpotent networks) [16]. As such it seems difficult to characterise the graphs where the disjunctive network minimises the periodic rank. For fixed points, on the other hand, the problem is straightforward. If $D$ is acyclic, then all the Boolean networks on $D$ have a unique fixed point by Robert's celebrated theorem [34]. Conversely, Aracena and Salinas (private communication) showed that any non-acyclic graph $D$ admits a fixed point free Boolean network.

## 4.2 Generalisations

We mention three possible avenues of generalising the scope of the current study of disjunctive networks. For each, it would be interesting to see how the results presented here might generalise to these broader classes of networks.

**Other Boolean networks** There are many classes of Boolean networks that contain, or are closely related to, disjunctive networks, e.g. AND-OR networks [2], AND-OR-NOT networks [3], nested canalyzing networks [29], threshold networks [19, 20], and of course monotone networks [5].

**Higher alphabets** The disjunction can be easily generalised to variables taking their values over a linearly ordered alphabet, by taking the maximum [1]. This is not the only choice that maintains some of the desirable properties of Boolean disjunction; in fact, a thorough study and classification of disjunction functions in multiple valued logics is given in [23]. A different approach views the conjunction as the product: $x \wedge y = xy$ for all $x, y \in \mathbb{B}$ [12]. One can then consider so-called monomial dynamical systems over finite fields [13], where each local function is a product of variables; note that those networks are not monotone.

**Infinite graphs** One can define the disjunction of any set of Boolean variables, whether finite or infinite. Considering disjunctive networks over infinite graphs would be another step to link Boolean networks and Cellular automata.

## References


1 J.A. Aledo, S. Martínez, F.L. Pelayo, and Jose C. Valverde. Parallel discrete dynamical systems on maxterm and minterm boolean functions. *Mathematical and Computer Modelling*, 55:666–671, 2012.

2 J. Aracena, J. Demongeot, and Eric Goles. Fixed points and maximal independent sets in AND-OR networks. *Discrete Applied Mathematics*, 138:277–288, 2004.

3 J. Aracena, A. Richard, and L. Salinas. Maximum number of fixed points in and-or-not networks. *Journal of Computer and System Sciences*, 80(7):1175–1190, 2014. doi:10.1016/j.jcss.2014.04.025.

4 Julio Aracena. Maximum number of fixed points in regulatory Boolean networks. *Bulletin of mathematical biology*, 70:1398–1409, 2008.

5 Julio Aracena, Jacques Demongeot, and Eric Goles. On limit cycles of monotone functions with symmetric connection graph. *Theoretical Computer Science*, 322:237–244, 2004.

6 Julio Aracena, Jacques Demongeot, and Eric Goles. Positive and negative circuits in discrete neural networks. *IEEE Transactions on Neural Networks*, 15(1):77–83, January 2004.

7 Julio Aracena, Adrien Richard, and Lilian Salinas. Number of fixed points and disjoint cycles in monotone boolean networks. *SIAM journal on Discrete mathematics*, 31(3):1702–1725, 2017.

8 Jorgen Bang-Jensen and Gregory Gutin. *Digraphs: Theory, Algorithms and Applications*. Springer, 2009.

9 Endre Boros. *Boolean functions*, chapter Horn functions, pages 269–321. Cambridge University Press, 2011.

10 Kim Ki-Hang Butler. The number of idempotents in $(0, 1)$-matrix semigroups. *Linear Algebra and its Applications*, 5:233–246, 1972.

11 Peter J. Cameron. *Combinatorics: Topics, Techniques, Algorithms*. Cambridge University Press, Cambridge, UK, 1994.

12 O. Colón-Reyes, R. Laubenbacher, and B. Pareigis. Boolean monomial dynamical systems. *Annals of Combinatorics*, 8(4):425–439, 2005.

13 Omar Colón-Reyes, Abdul Salam Jarrah, Reinhard Laubenbacher, and Bernd Sturmfels. Monomial dynamical systems over finite fields. *Complex Systems*, 16:333–342, 2006.

**14** Maximilien Gadouleau. On the rank and periodic rank of finite dynamical systems. *Electronic Journal of Combinatorics*, 25(3):1–16, 2018.

**15** Maximilien Gadouleau. On the influence of the interaction graph on a finite dynamical system. *Natural Computing*, 19:15–28, 2020.

**16** Maximilien Gadouleau and Adrien Richard. Simple dynamics on graphs. *Theoretical Computer Science*, 628:62–77, 2016.

**17** Maximilien Gadouleau, Adrien Richard, and Søren Riis. Fixed points of boolean networks, guessing graphs, and coding theory. *SIAM Journal on Discrete Mathematics*, 29(4):2312–2335, 2015.

**18** E. Goles and M. Noual. Disjunctive networks and update schedules. *Advances in Applied Mathematics*, 48(5):646–662, 2012.

**19** E. Goles and J. Olivos. Comportement périodique des fonctions à seuil binaires et applications. *Discrete Applied Mathematics*, 3:93–105, 1981.

**20** Eric Goles. Dynamics of positive automata networks. *Theoretical Computer Science*, 41:19–32, 1985.

**21** Eric Goles and Gonzalo Hernández. Dynamical behavior of Kauffman networks with AND-OR gates. *Journal of Biological Systems*, 8(2):151–175, 2000.

**22** Eric Goles and M. Tchuente. Iterative behaviour of generalized majority functions. *Mathematical Social Sciences*, 4:197–204, 1983.

**23** Siegfried Gottwald. *A Treatise on Many-Valued Logics*. Research Studies Press Ltd., Baldock, Hertfordshire, England, 2001.

**24** Karl-Peter Hadeler and Johannes Müller. *Cellular Automata: Analysis and Applications*. Springer, Cham, Switzerland, 2017.

**25** Lisa Hellerstein. *Boolean functions*, chapter Characterizations of special classes by functional equations, pages 487–506. Cambridge University Press, 2011.

**26** John M. Howie. *Fundamentals of Semigroup Theory*. Oxford Science Publications, 1995.

**27** Andrew Ilachinski. *Cellular Automata: A Discrete Universe*. World Scientific, Singapore, 2001.

**28** A. Salam Jarrah, R. Laubenbacher, and A. Veliz-Cuba. The dynamics of conjunctive and disjunctive boolean network models. *Bulletin of Mathematical Biology*, 72:1425–1447, 2010.

**29** S. Kauffman, C. Peterson, B. Samuelsson, and C. Troein. Genetic networks with canalyzing boolean rules are always stable. *Proceedings of the National Academy of Sciences of the United States of America*, 101:17102–17107, 2004.

**30** Ki Hang Kim. *Boolean Matrix Theory and Applications*. Marcel Dekker, Inc., New York, 1982.

**31** Loïc Paulevé and Adrien Richard. Static analysis of boolean networks based on interaction graphs: A survey. *Electronic Notes in Theoretical Computer Science*, 284:93–104, 2012.

**32** Reinhard Pöschel and Ivo Rosenberg. *Boolean models and methods in mathematics, computer science, and engineering*, chapter Compositions and clones of Boolean functions, pages 3–38. Cambridge University Press, 2010.

**33** Søren Riis. Utilising public information in network coding. In *General Theory of Information Transfer and Combinatorics*, volume 4123/2006 of *Lecture Notes in Computer Science*, pages 866–897. Springer, 2006.

**34** F. Robert. Iterations sur des ensembles finis et automates cellulaires contractants. *Linear Algebra and its Applications*, 29:393–412, 1980.

**35** David Rosenblatt. On the graphs of finite idempotent boolean relation matrices. *Journal of Research of the National Bureau of Standards–B. Mathematics and Mathematical Physics*, 67B(4):259–256, October–December 1963.

# Regular Languages: To Finite Automata and Beyond

## Succinct Descriptions and Optimal Simulations

**Luca Prigioniero**
Dipartimento di Informatica, Università degli Studi di Milano, Milan, Italy

**Abstract**

It is well known that the class of regular languages coincides with the class of languages recognized by finite automata. Nevertheless, many other characterizations of this class in terms of computational devices and generative models are present in the literature. For example, by suitably restricting more powerful models such as context-free grammars, pushdown automata, and Turing machines, it is possible to obtain formal models that generate or recognize regular languages only. These restricted formalisms provide alternative representations of regular languages that may be significantly more concise than other models that share the same expressive power.

The goal of this work is to provide an overview of old and recent results on these formal systems from a descriptional complexity perspective, that is by considering the relationships between the sizes of such devices. We also present some results related to the investigation of the famous question posed by Sakoda and Sipser in 1978, concerning the size blowups from nondeterministic finite automata to two-way deterministic finite automata.

**2012 ACM Subject Classification** Theory of computation → Regular languages; Theory of computation → Models of computation

**Keywords and phrases** Regular languages, descriptional complexity, finite automata

**Digital Object Identifier** 10.4230/OASIcs.AUTOMATA.2021.2

**Category** Invited Talk

**Acknowledgements** This paper and the research behind it would not have been possible without the support and the valuable comments of Giovanni Pighizzini.

## 1 Introduction

The investigation of computational models operating under restrictions is one of classical topics of computer science.

In one of his pioneer papers, Chomsky introduced a hierarchy of classes of languages, also known as *Chomsky hierarchy*, obtained by applying increasing restrictions to general *grammars*, that characterize the class of *recursively enumerable* languages [7]. In this way he introduced the classes of *context-sensitive*, *context-free*, and *regular* languages.

For the same classes of languages, there also exist characterizations in terms of *computational devices*. Even in this case, bounding computational resources of general models, less powerful devices can be obtained. For example, while Turing machines (TM for short), even in the variant with one tape only, characterize recursively enumerable languages, by restricting the working space they are allowed to use to the portion of the tape that initially contains the input, linear bounded automata are obtained, that are equivalent to context-sensitive grammars [33]. Also finite automata and pushdown automata (PDA), that are standard recognizers for regular and context-free languages, respectively, can be considered as particular Turing machines in which the capacity or access to the memory storage is limited.

27th IFIP WG 1.5 International Workshop on Cellular Automata and Discrete Complex Systems (AUTOMATA 2021).
Editors: Alonso Castillo-Ramirez, Pierre Guillon, and Kévin Perrot; Article No. 2; pp. 2:1–2:16

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Besides the standard models mentioned thus far, considering machines that make a restricted use of resources, it is possible to obtain alternative characterizations of the classes of the hierarchy. For example, Hennie proved that when the length of the computations, *i.e.*, the time, is linear in the input length, one-tape Turing machines are no more powerful than finite automata, that is, they recognize regular languages only [22].

As remarked by Chomsky, context-free languages have the property of being able to describe recursive structures such as, for instance, nested parentheses, arithmetic expressions, and typical programming language constructs. In terms of recognizing devices, this capability is typically implemented through the pushdown store, a memory structure in which the information is stored and recovered in a "last in–first out" way, which adds recursion to finite automata, so making the resulting model (pushdown automata), equivalent to context-free grammars (CFG) [8].

To emphasize the ability of context-free grammars to generate recursive sentential forms, Chomsky investigated the *self-embedding* property: a context-free grammar is self-embedding if it contains some variable which, in some sentential form, is able to reproduce itself enclosed between two nonempty strings [6]. Roughly speaking, this means that such a *self-embedded* variable can generate a "true" recursion that needs an auxiliary memory (typically a pushdown) to be implemented (in contrast with tail or head recursions, corresponding to the cases in which the two strings surrounding the variable are empty, that can be easily eliminated). Chomsky proved that, among all context-free grammars, only self-embedding ones can generate nonregular languages. Hence, *non-self-embedding grammars* (NSE) are no more powerful than finite automata [7, 6].

Counterpart devices for non-self-embedding grammars, for which the capability of recognizing recursive structures is limited by placing some restrictions on the size of the memory of the corresponding general model, are *constant-height pushdown automata* ($h$-PDA). More precisely, these devices are standard nondeterministic pushdown automata where the amount of available pushdown store is bounded by some constant $h \in \mathbb{N}$. Hence, the number of their possible configurations is finite, thus implying that they are no more powerful than finite automata.

By contrast to models that make use of space or time restrictions, Hibbard introduced *$d$-limited automata* ($d$-LA), that are obtained by limiting the writing capabilities of linear bounded automata allowing overwriting of each tape cell only the first $d$ times that it is scanned, for some fixed $d \geq 0$ [23]. Nevertheless, the cell may be visited again and the information stored therein read arbitrarily many more times, but its contents is *frozen* for the remainder of the computation. Hibbard proved that, for each $d \geq 2$ this model characterizes context-free languages and showed the existence of an infinite hierarchy of deterministic $d$-limited automata, whose first level (*i.e.*, corresponding to deterministic 2-limited automata) has been later proved to coincide with the class of *deterministic context-free languages* [43]. (See [34] and references therein for further connections between limited automata and context-free languages.) Furthermore, as shown by Wagner and Wechsung, when $d = 1$, that is, when these devices are allowed to overwrite the contents of each tape cell during the first visit only, 1-limited automata (1-LA) are equivalent to finite automata [56]. It is a trivial observation that even when $d = 0$, and hence no writings on the tape are allowed, the model still recognizes regular languages only: 0-limited automata correspond to finite automata that can move their input head back and forth on the input tape, namely *two-way finite automata*.[1]

[1] Further technical details, properties, examples, and references about limited automata can be found in the recent survey by Pighizzini [41].

General devices and their restrictions characterizing regular languages and discussed in this work are depicted in Figure 1.

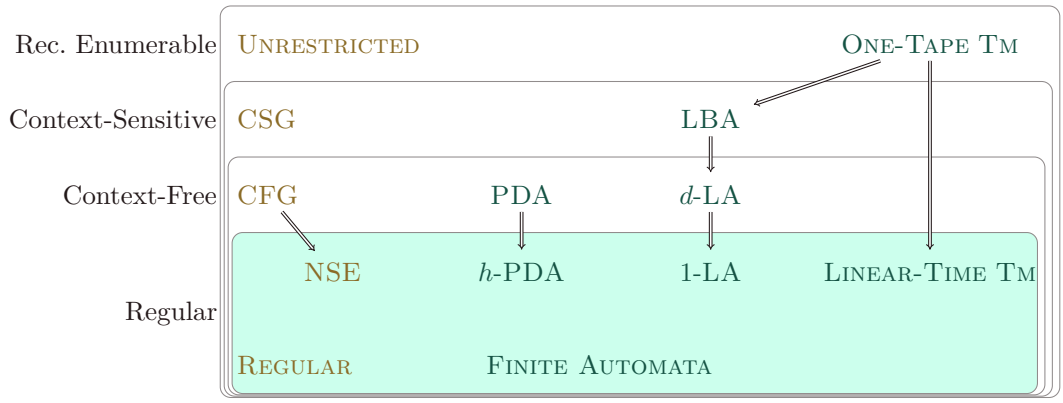



**Figure 1** Models representing regular languages obtained by posing some restrictions on standard grammars (on the left, in gold) and recognizers (on the right, in green) characterizing the classes of the Chomsky hierarchy.

## Descriptional Complexity of Formal Systems

In this work we shall focus on the models characterizing the bottom level of the Chomsky hierarchy: the class of regular languages. We give an overview of some recent results obtained in the area of descriptional complexity, specifically, the study of the capability of these devices of representing the same class of languages in a more, or less, concise way. More precisely, *descriptional complexity* is a branch of theoretical computer science whose goal is the investigation of the relationships between the sizes of the representations of formal systems that share the same computational power, or, in other words, the study of how concisely a system can describe a class of problems (or languages).

The devices characterizing the class of regular languages introduced at the beginning of the section are obtained by limiting some resources of more general models. Therefore, the main question we are interested about is *"How much does the limitation of one resource cost in terms of another resource, or, in other words, what are the upper and lower bounds of such trade-offs?"* [14].

Since our goal is comparing the sizes of the descriptions of devices and formal systems, for each model under consideration we evaluate its *size* as the total number of symbols used to describe it, or, in other words, to write down its description. In particular, to measure the size of recognizing devices, we consider the amount of memory required to store the "*algorithm*" they implement (their transition function), so, for example, the size of finite automata is bounded by a polynomial in the number of their states. On the other hand, the size of grammars is given by the number of symbols used to write down their derivation rules [32].

Given two different classes of computational models $\mathcal{M}_1$ and $\mathcal{M}_2$ characterizing regular languages, there are natural questions we are interested in:

- Does a function $F$ exist, such that for all the regular languages $L$, the size of the smallest device of type $\mathcal{M}_1$ for $L$ is upper bounded by the function $F$ of the size of the smallest equivalent device of type $\mathcal{M}_2$? If $F$ exists, it is an *upper bound* for the increase (*blow-up*)

in complexity when changing from a minimal model of type $\mathcal{M}_2$ for an arbitrary regular language to an equivalent minimal model of type $\mathcal{M}_1$.

- Do an infinite family of distinct regular languages $L_i$ and a function $f$ exist, such that for all $i \in \mathbb{N}$, the size of any device of type $\mathcal{M}_1$ for $L_i$ is lower bounded by the function $f$ of the size of the smallest equivalent device of type $\mathcal{M}_2$? If $f$ exists, it is a *lower bound* for the increase in complexity when changing from a minimal model of type $\mathcal{M}_2$ to an equivalent minimal model of type $\mathcal{M}_1$ for infinitely many languages.

If there is no recursive function upper bounding the trade-off between two computational models $\mathcal{M}_1$ and $\mathcal{M}_2$, the trade-off is *non-recursive*. Furthermore, if the lower and the upper bounds coincide, we say that the bound is *optimal*. For more details about the area of descriptional complexity see, *e.g.*, the surveys by Goldstine et al., and Holzer and Kutrib [14, 24].

A classical problem in this field is the investigation of the relationships between deterministic and nondeterministic devices. It is well known that *one-way deterministic* finite automata (1DFA) are sufficient for regular languages. By allowing nondeterministic transitions, thus obtaining one-way nondeterministic finite automata (1NFA), the computational power does not increase [51]. A natural question concerning models that share the same computational power is the comparison of their size.

As an example, consider, for any fixed integer $n \geq 0$, the language $L_n = (0+1)^*1(0+1)^{n-1}$, composed by strings on the alphabet $\{0, 1\}$ whose $n$-th to last symbol is 1. A 1NFA $\mathcal{A}_n$ could recognize $L_n$ by moving the input head forward on the input tape, until reaching the $n$-th symbol from the end, that is detected by performing a nondeterministic guess. If such a symbol is 1, the automaton checks whether the length of the remaining part of the input string is $n-1$. Therefore, it is easy to check that the number of states for a 1NFA accepting $L_n$ is $n+1$. On the other hand, a 1DFA accepting $L_n$ cannot guess which is the $n$-th symbol from the end of the input string. So, intuitively, it has to "*remember*" a factor representing the last $n$ symbols read, by storing it in the finite control. When the end of the input is reached, the device checks that the leftmost symbol of the current factor is 1. It is easy to see that the number of the possible factors of length $n$ is $2^n$, and each of them is stored by using a state of the 1DFA, thus implying an exponential blowup in states with respect to the equivalent 1NFA $\mathcal{A}_n$.

Therefore, even if deterministic and nondeterministic finite automata characterize the same class of languages, one-way deterministic automata can require exponentially many states with respect to equivalent nondeterministic automata. Hence, there is an exponential size gap from one-way nondeterministic to one-way deterministic automata.

It is also known that even providing finite automata with the capability of moving the input head back and forth along the tape, thus obtaining *two-way* finite automata (2DFA and 2NFA), their recognizing power does not increase. In fact, Shepherdson and, independently, Rabin and Scott, proved this result by giving two constructions based on the analysis of the moves of the input head of the automata between the tape cells [51, 53]. Both constructions, given a two-way finite automaton, return an equivalent one-way deterministic finite automaton whose size is exponential in the square of the size of the automaton given in input.

At this point, one could ask *"Is it possible to exploit the ability to scan the input in a two-way fashion in finite automata to eliminate the nondeterminism?"*. Differently from the one-way case, the question about the size cost of the conversion of (one-way and two-way) nondeterministic finite automata into two-way deterministic finite automata, that was posed by Sakoda and Sipser, is open since 1978. In their paper, Sakoda and Sipser formulated the problem by the questions

1. For every 2NFA $\mathcal{A}$, does an equivalent 2DFA $\mathcal{A}'$ exist, such that $\mathcal{A}'$ has only polynomially more states than $\mathcal{A}$?
2. For every 1NFA $\mathcal{A}$, does an equivalent 2DFA $\mathcal{A}'$ exist, such that $\mathcal{A}'$ has only polynomially more states than $\mathcal{A}$?

They conjectured that, in both cases, the answer is negative and an exponential increase of the size is necessary.

The question has been solved in some special cases that can be grouped in three classes: by considering restrictions on the simulating machines (*e.g.*, *sweeping* [54], *oblivious* [27], *few reversals* two-way deterministic finite automata [28]), by considering restrictions on languages (*e.g.*, *unary case* [12]), by considering restrictions on the simulated machines (*e.g.*, *outer-nondeterministic* automata [10, 30]). However, in spite of all attempts, in the general case the question remains open.

The importance of this open problem is supported by its similarity to the well-known $\mathrm{P} \stackrel{?}{=} \mathrm{NP}$ problem [52] and by relationships with the $\textsc{LogSpace} \stackrel{?}{=} \textsc{NLogSpace}$ question [5, 13, 29, 30]. (See [40] for further details and references.)

## Outline

This work is an excerpt of the dissertation [48] and an informal version appeared in [49]. It is organized as follows (a summary of some of the main results obtained is depicted in Figure 2).

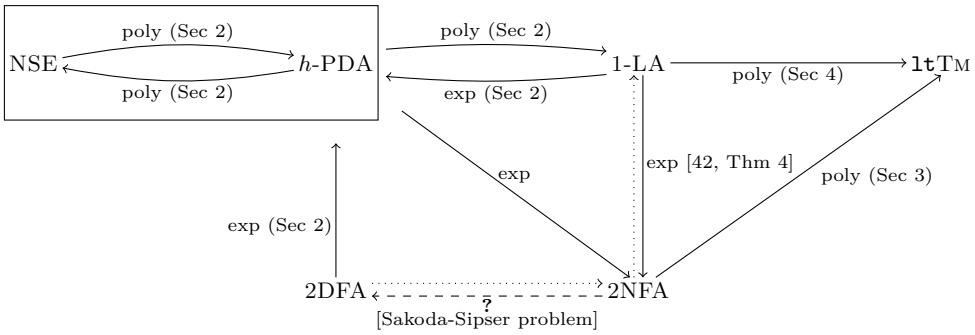



**Figure 2** Some of the main results discussed in the work. Dotted arrows denote trivial relationships, while the dashed arrow indicates the Sakoda and Sipser's question [52]. Linear-time Turing machines are denoted ltTM.

Section 2 presents some results on non-self-embedding grammars and their descriptional complexity [46], and their relations in size with constant-height pushdown automata and 1-limited automata [15].

Section 3 investigates machines one-tape Turing machines working in linear time. It is not decidable whether a one-tape Turing machine works in linear time, even if it is deterministic and restricted to use only the portion of the tape which initially contains the input, unless the machine is *weight-reducing*, *i.e.*, syntactically forced to operate in linear-time [16]. By relating the study of Turing machines working in linear time to the above-mentioned open question of Sakoda and Sipser, we present the costs of the conversion of nondeterministic finite automata into equivalent linear-time one-tape deterministic machines. A polynomial blowup from two-way nondeterministic finite automata into equivalent weight-reducing one-tape deterministic machines (that work in linear time) has been obtained.

The investigation about devices working in linear time is continued in Section 4, where the time complexity of 1-limited automata is analyzed from a descriptional complexity point of view [19]. Though the model recognizes regular languages only, it may use quadratic time in the input length. With a polynomial increase in size and preserving determinism, each 1-limited automaton can be transformed into a linear-time equivalent one. We also obtained polynomial transformations into linear-time Turing machines.

Section 5 turns the attention to pushdown automata [45]. In particular, we studied pushdown automata without any restriction on the pushdown height and we showed that it cannot be decided whether these devices accept using constant pushdown height, with respect to the input length, or not. Furthermore, in the case of acceptance in constant height, the height cannot be bounded by any recursive function in the size of the description of the machine. In contrast, in the restricted case of pushdown automata over a one-letter input alphabet, *i.e.*, unary pushdown automata, the above property becomes decidable. Moreover, if the height is bounded by a constant that does not depend on the input length, then it is at most exponential with respect to the size of the description of the pushdown automaton. This bound cannot be reduced. Finally, if a unary pushdown automaton uses non-constant height to accept, then the height should grow at least as the logarithm of the input length. This bound is optimal.

In conclusion, in Section 6 we briefly discuss some possible future research directions and open problems.

## 2 Non-Self-Embedding Grammars, Constant-Height Pushdown Automata, and Limited Automata

As mentioned in Section 1, Chomsky investigated the self-embedding property of context-free grammars, and proved that non-self-embedding grammars only generate regular languages. The proof of the result given by Chomsky is constructive: it provides a method for obtaining a finite automaton equivalent to a given non-self-embedding grammar [7, 6]. A different constructive proof of the same result was given by Anselmo, Giammarresi, and Varricchio, who showed that it is possible to decompose non-self-embedding grammars into regular grammars and then to iteratively apply regular substitutions in order to obtain equivalent finite automata [2]. In the same paper, the authors also presented an NSE grammar for the following family of languages. For any fixed integer $n$, let $U_n = \{ a^{2^n} \}$ be the singleton composed by the unary string of length $2^n$. $U_n$ can be generated by an NSE grammar with variables $\{ A_0, \ldots, A_n \}$ such that, for each $i = 1, \ldots, n$, the variable $A_i$ generates two occurrences of the variable $A_{i-1}$, and the variable $A_0$ produces the terminal $a$. It is possible to see that the variable $A_i$ derives the string $a^{2^i}$, for $i = 0, \ldots, n$. Hence, the language generated is $U_n$. Moreover, the size of the grammar is linear in the parameter $n$, while it is not hard to verify that the equivalent minimum deterministic finite automaton has $2^n + 1$ states. Actually, as a consequence of state lower bound presented by Mereghetti and Pighizzini, the same amount of states is also necessary to accept $U_n$ by using a 2NFA [36]. Therefore, this language witnesses that the size gap between non-self-embedding grammars and equivalent finite automata is at least exponential.

It is worthwhile to mention that, as proved by Meyer and Fischer, for any recursive function $f$ and arbitrarily large integer $n$, there exists a context-free grammar whose description has size $n$, the generated language is regular, and any equivalent finite automaton requires at least $f(n)$ states [37]. This means that it is not possible to obtain a recursive bound relating the size of context-free grammars generating regular languages with the number of states of equivalent finite automata. It is important to notice that the result by Meyer and Fischer

was obtained by considering grammars with a two-letter terminal alphabet. The unary case was studied by Pighizzini, Shallit, and Wang, who obtained optimal recursive bounds [47].

We recently proved that also in the case of non-self-embedding grammars the bounds are recursive, independently on the alphabet size [46]. In particular, by inspecting and refining the construction presented by Anselmo, Giammarresi, and Varricchio, we showed that

▶ **Result 2.1** ([46]). *Each non-self-embedding grammar of size $n$ can be converted into equivalent nondeterministic and deterministic automata with $2^{O(n)}$ and $2^{2^{O(n)}}$ states, respectively.*

We also obtained a family of languages that witnesses that these gaps cannot be reduced. Furthermore, these gaps do not change if we allow the variables generating only unary strings (*i.e.*, strings consisting of occurrences of only one terminal) to be self-embedded.

Other formal models characterizing the class of regular languages and exhibiting gaps of the same order with respect to deterministic and nondeterministic automata are constant-height pushdown automata and 1-limited automata. So, it is natural to study the size relationships between non-self-embedding grammars, constant-height pushdown automata, and 1-limited automata, three models that restrict context-free acceptors to the level of regular recognizers.

The exponential and double exponential gaps from constant-height pushdown automata to nondeterministic and deterministic automata have been proven by Geffert, Mereghetti, and Palano [11]. Furthermore, Bednárová et al. showed the interesting result that the gap from nondeterministic to deterministic constant-height pushdown automata is double exponential also [3]. We remind the reader that both non-self-embedding grammars and constant-height pushdown automata are restrictions of the corresponding general models, where true recursions are not possible. By adapting the standard transformation from PDA to CFGs, and by modifying the decomposition of NSE grammars presented by Anselmo, Giammarresi, and Varricchio, we proved that

▶ **Result 2.2** ([15]). *Non-self-embedding grammars and constant-height pushdown automata are polynomially related in size.*

Also 1-limited automata can be significantly smaller than equivalent finite automata. The equivalence between 1-limited automata and finite automata has been investigated from the descriptional complexity point of view by Pighizzini and Pisoni, who proved that each 1-limited automaton $\mathcal{A}$ with $n$ states can be simulated by a one-way deterministic automaton with a number of states doubly exponential in a polynomial in $n$. Furthermore, in the worst case, double exponentially many states are necessary for this simulation. The cost reduces to a single exponential when $\mathcal{A}$ is deterministic [42]. The lower bounds in this result have been obtained by providing witness languages defined over a binary alphabet.

We investigated the unary case, for which it was an open question whether the same bounds held. In particular, we obtained that

▶ **Result 2.3** ([44]). *There is an exponential gap between unary deterministic 1-limited automata and two-way nondeterministic finite automata.*

To this end, we showed that, for each $n > 1$, the singleton language $U_n = \{ a^{2^n} \}$ can be recognized by a deterministic 1-limited automaton having $2n + 1$ states and a description of size $O(n^2)$. Since the same language requires $2^n + 1$ states to be accepted by a one-way nondeterministic automaton, it turns out that the state gap between deterministic 1-limited automata and one-way nondeterministic automata in the unary case is the same as in the binary case. It is an easy observation that the gap does not reduce if we want to convert unary deterministic 1-limited automata into two-way nondeterministic automata.

We also considered the relationships between 1-limited automata and unary context-free grammars. The cost of the conversion of these grammars into finite automata has been investigated: exponential gaps have been proven by Pighizzini, Shallit, and Wang [47]. With the help of a result presented by Okhotin [38], we proved that

▶ **Result 2.4** ([44]). *Each unary context-free grammar $\mathcal{G}$ can be converted into an equivalent* 1*-limited automaton whose description has a size that is polynomial in the size of $\mathcal{G}$.*

Let us now turn our attention to the size relationships between 1-limited automata and non-self-embedding grammars.

We obtained a construction transforming each non-self-embedding grammar into a 1-limited automaton of polynomial size. In particular, given an input $w$, the 1-LA nondeterministically generates a *compression* of a derivation tree of $w$. Then, it verifies the validity of such a guess. As a consequence,

▶ **Result 2.5** ([15]). *Each constant-height pushdown automaton can be transformed into an equivalent* 1*-limited automaton of polynomial size.*

Even the conversion of deterministic constant-height pushdown automata into deterministic 1-limited automata costs polynomial in size. For the converse transformation, we showed that an exponential size is necessary. Indeed, consider the following family of languages. For any fixed integer $n > 0$, let $P_n$ be the language of the powers of any string $u$ of length $n$ over the alphabet $\{0,1\}$, *i.e.*, $P_n = \{ u^k \mid u \in \{0,1\}^n,\ k \geq 0 \}$. $P_n$ can be accepted by a two-way deterministic finite automaton (and, hence, by a 1-LA) with $O(n)$ states, but requires exponentially many states to be accepted even by an unrestricted pushdown automaton, which is forced to store the word $u$ in its finite control. From the cost of the conversion of 1-limited automata into nondeterministic automata, it turns out that for the conversion of 1-limited automata into non-self-embedding grammars an exponential size is also sufficient.

▶ **Result 2.6** ([15]). *The size cost of the conversion of* 1*-limited automata into non-self-embedding grammars and constant-height pushdown automata is exponential.*

Bednárová et al. raised the question of the cost of the conversion of deterministic $h$-PDA into 1NFAs [3]. To this regard, it is possible to observe that, for any integer $n \geq 0$, the language $S_n = \left(a^{2^n}\right)^*$ is accepted by a deterministic $h$-PDA of size polynomial in $n$ (a constant-height pushdown automaton for $S_n$ with a constant number of states, $h = n$, and $2n+1$ pushdown symbols can be found in [45]) but, as discussed at the beginning of this section, it requires an exponential number of states to be accepted by a finite automaton. Hence, we can conclude that the cost of both simulations from two-way automata to $h$-PDA and from $h$-PDA to two-way automata is at least exponential.

More details about the results presented in this section can be found in [15, 44, 46].

## 3 Two-Way Automata and One-Tape Machines

In this section we present some results on one-tape Turing machines working in linear time and on the relationships between the sizes of their descriptions with those of equivalent finite automata. We consider the following variants of one-tape *deterministic* Turing machines:

**End-marked machines.** We say that a device is *end-marked* if each input string is surrounded by two special symbols called the left and the right endmarkers and the machine is restricted to use only the portion of the tape that initially contains the input (plus the endmarkers), that cannot be left by the head.

**Weight-reducing Turing machines.** Roughly speaking, in *weight-reducing* Turing machines each overwriting is decreasing with respect to some fixed order on the working alphabet. As a consequence, after overwriting a cell with a minimal symbol, such a machine cannot visit the cell again. By this condition, in weight-reducing Turing machines the number of visits to each tape cell is bounded by a constant. However, weight-reducing machines could have non-halting computations which, hence, visit infinitely many tape cells.

**Linear-time Turing machines.** A deterministic Turing machine is said to be *linear-time* if, over each input $w$, its computation halts within $O(|w|)$ steps.

**Hennie machines.** A Hennie machine is a linear-time Turing machine which is, furthermore, end-marked.

**Weight-Reducing Hennie machines.** These devices are defined by combining previous conditions. Observe that each end-marked weight-reducing Turing machine can execute a number of steps which is at most linear in the length of the input. Consequently, end-marked weight-reducing deterministic Turing machines are necessarily Hennie machines.

It is useful to recall that it cannot be decided whether or not a one-tape Turing machine works in linear time.[2] Furthermore, there is no recursive function bounding the size blowup from one-tape Turing machines working in linear time to equivalent finite automata. We proved that these results remain true even in the restricted case of end-marked machines.

▶ **Result 3.1** ([16, 18]). *It is undecidable whether an end-marked Turing machine works in linear time.*

To overcome the above-mentioned "negative" results, we considered weight-reducing machines, that can be seen as a syntactical restriction of one-tape Turing machines. In this case, it can be decided whether a deterministic Turing machine is weight reducing. These devices can have non-halting computations. However, they work in linear time as soon as they are halting. In fact, we showed that it is possible to decide whether a weight-reducing machine is halting. As a consequence,

▶ **Result 3.2** ([16, 18]). *It is possible to decide whether a weight-reducing machine works in linear time.*

Moreover, by a polynomial size increase, each weight-reducing machine can be transformed into an equivalent one which always halts and works in linear time. The same blowup is easily extended to weight-reducing Hennie machines.

▶ **Result 3.3** ([16, 18]). *With a polynomial size increase, any weight-reducing machine can be made halting, so working in linear time. Furthermore, the cost of the simulation of weight-reducing machines by* 1DFA *is doubly exponential.*

Furthermore, we proved that

▶ **Result 3.4** ([16, 18]). *Each linear-time machine $\mathcal{T}$ can be converted into an equivalent weight-reducing one whose size is bounded by a polynomial function in the size and in the execution time of $\mathcal{T}$.*

We also related the study of these restricted variants of Turing machines accepting only regular languages to the Sakoda and Sipser question. In this case, we propose a new approach

[2] We mention that it is decidable, though, whether or not a machine makes at most $cm + d$ steps on input of length $m$, for any *fixed* $c, d > 0$ [9].

in which we considered, as target devices, linear-time one-tape *deterministic* Turing machines. It turned out that each 2NFA $\mathcal{A}$ can be simulated by a one-tape deterministic Turing machine which works in linear time (with respect to the input length) and which has a polynomial size with respect to the size of $\mathcal{A}$. We point out that the resulting machine can use extra space, besides the tape segment which initially contained the input. Next, the machine is halting and weight reducing, thus implying a linear execution time. Hence,

▶ **Result 3.5** ([16, 17]). *Nondeterminism can be eliminated from* 2NFA *with at most a polynomial size increase, obtaining a linear execution time in the input length, provided the ability to rewrite tape cells and to use some extra space.*

We then investigated what happens by removing the latter possibility, namely if the machine does not have any further tape storage, *i.e.*, it is a Hennie machine. We proved that even under this restriction it is still possible to obtain a machine of polynomial size, namely

▶ **Result 3.6** ([16, 17]). *Each* 2NFA *can be transformed into an equivalent Hennie machine of polynomial size.*

However, the machine resulting from our construction is not weight reducing, unless we require that it agrees with the given 2NFA only on sufficiently long inputs. This problem does not occur in the unary case, where we proved that

▶ **Result 3.7** ([16, 17]). *Each unary* 2NFA *can be simulated by a weight-reducing Hennie machine of polynomial size.*

Similar results have been obtained for the transformation of 1NFAs into variants of one-tape deterministic machines.

For details about the results discussed in this section we address the reader to [16, 17, 18].

## 4 Limited Automata: A Time Constraint

As proved by Hennie, deterministic one-tape Turing machines operating in linear time recognize exactly the class of regular languages [22]. The result has later been extended to the nondeterministic case by Pighizzini and Tadaki, Yamakami, and Lin [39, 55]. Here, *operating in linear time* means that every computation has length linearly bounded in the input length. In particular, linear-time machines are necessarily halting – see [39] for investigations of alternative linear-time restrictions. The above-mentioned result implies that every Hennie machine is equivalent to some finite automaton. From the opposite point of view, this means that providing two-way finite automata with the ability to overwrite the tape cells does not extend the expressiveness of the model, as long as the time is linearly bounded in the length of the input.

However, it is undecidable, given an end-marked deterministic Turing machine (*i.e.*, a deterministic linear bounded automaton), to check whether it works in linear time over all input strings, or, in other words, whether it is actually a deterministic Hennie machine [16, 17, 50]. To avoid this drawback, Průša proposed the weight-reducing variant of Hennie machines, in which the time limitation is syntactic. As a consequence, the number of visits of a cell by the head is bounded by some constant (*i.e.*, not depending on the input length), hence the device works in linear time over every input string.

By contrast to Hennie machines, in $d$-limited automata the head is allowed to visit a cell after the $d$-th visit, even if it cannot rewrite the contents anymore. This allows to use super-linear time.

As an example let us consider, for each fixed integer $n \geq 0$, the language

$$Q_n = \{\, x_0 x_1 \cdots x_k \mid k > 0, \text{ for each } i\colon\ x_i \in \Sigma^n \text{ and for some } j \neq 0\colon\ x_j = x_0 \,\}$$

on the alphabet $\Sigma = \{\, 0, 1 \,\}$. A deterministic 1-LA $\mathcal{A}_n$ may recognize $Q_n$ as follows. It first scans the factor $x_0$, overwriting each input symbol with a marked copy. Then, $\mathcal{A}_n$ repeats a subroutine which overwrites a factor $x_j$ with the marker $\sharp \notin \Sigma$, while checking whether $x_j$ equals $x_0$ or not. This can be achieved as follows. Before overwriting the $\ell$-th symbol of $x_j$, first, $\mathcal{A}_n$, with the help of a counter modulo $n$, moves the head leftward to the position $\ell$ of $x_0$ and stores the unmarked scanned symbol $\sigma$ in its finite control; second, it moves the head rightward until reaching the position $\ell$ of $x_j$, namely, the leftmost position that has not been overwritten so far. At this point, $\mathcal{A}_n$ compares the scanned symbol (*i.e.*, the $\ell$-th symbol of $x_j$) with $\sigma$ (*i.e.*, the $\ell$-th symbol of $x_0$). When $\mathcal{A}_n$ finds out that a complete factor $x_j$ matches $x_0$, it reaches the end of the input checking that has length multiple of $n$.

It is possible to implement $\mathcal{A}_n$ with a number of states linear in $n$ and $\#\Sigma{+}1$ working symbols. Since for each position of a factor $x_i$, $i > 0$, the head has to move back to the factor $x_0$, it is possible to observe that $\mathcal{A}_n$ works in quadratic time in the length of the input string.

Therefore, also in the case $d = 1$, $d$-limited automata can operate in super-linear time. This contrasts with Hennie machines which operate in linear time by definition. The question we addressed is whether this ability of 1-limited automata with respect to Hennie machines yields a gap between the two models in terms of the size of their representations.

We proved that operating in super-linear time is not essential for 1-LA, if allowing a polynomial increase in the number of states. In other words,

▶ **Result 4.1** ([19]). *With a polynomial increase in size, each* 1*-limited automaton can be transformed into an equivalent linear-time* 1*-limited automaton, or, alternatively, into a weight-reducing Hennie machine.*

Furthermore, the obtained device is deterministic when the original machine is deterministic as well. This is achieved by extending the exponential-cost simulation of 1-LA by 2NFA given in [42] (which in turn extends Shepherdson's classic conversion of 2DFA into 1DFA [53]) with a method for storing and accessing a carefully chosen subcollection of the many "Shepherdson tables" that a 1NFA would need to remember in its states: the simulating automaton can both store the tables (despite the 1-limitation) and access them efficiently (in both size and time).

For details about the results discussed in this section please refer to [19].

## 5 Pushdown Automata and Space Restrictions

As discussed in Section 2, constant-height pushdown automata, that are pushdown automata in which the maximum height of the pushdown is limited by some constant, allow more succinct representations of regular languages than finite automata [11], and are polynomially related in size with their natural generative counterpart, NSE grammars, roughly context-free grammars without "true" recursion [7].

In this section we turn our attention to standard pushdown automata, namely with an unrestricted pushdown store, that, however, are able to accept their inputs by making use only of a constant amount of the pushdown store. More precisely, we say that a pushdown automaton $\mathcal{M}$ *accepts in constant height* $h$, for some given $h$, if for each word in the language accepted by $\mathcal{M}$ there exists one accepting computation in which the maximum height reached

by the store is bounded by $h$. Notice that this does not prevent the existence of accepting or rejecting computations using an unbounded pushdown height.

It is a simple observation that a pushdown automaton $\mathcal{M}$ accepting in constant height $h$ can be converted into an equivalent constant-height pushdown automaton: in any configuration it is enough to keep track of the current height in order to stop and reject when a computation tries to exceed the height limit. The description of the resulting constant-height pushdown automaton has size polynomial in $h$ and in the size of the description of $\mathcal{M}$.

While studying these size relationships, we tried to understand *how large the height $h$ of the pushdown can be with respect to the size of the description of $\mathcal{M}$*. We discovered that $h$ can be arbitrarily large. Indeed, adapting an argument presented by Meyer and Fischer to prove non recursive trade-offs between the size of PDA accepting regular languages and the number of states of equivalent automata [37], we showed that

▶ **Result 5.1** ([45]). *There is no recursive function bounding the maximal height reached by the pushdown store in a pushdown automaton accepting in constant height, with respect to the size of its description.*

With the same argument, we obtained that

▶ **Result 5.2** ([45]). *There is no recursive function bounding the size blowup from PDA accepting in constant height to finite automata.*

Moreover, using a technique introduced by Hartmanis, based on suitable encodings of single-tape Turing machine computations [21], we also proved that

▶ **Result 5.3** ([45]). *It cannot be decided whether a pushdown automaton accepts in constant height or not.*

We also investigated the unary case. By studying the structure of the computations of unary pushdown automata, we were able to prove that, in contrast to the general case,

▶ **Result 5.4** ([45]). *It can be decided whether or not unary pushdown automata accept in constant height.*

Furthermore, we proved that

▶ **Result 5.5** ([45]). *If a unary pushdown automaton $\mathcal{M}$ accepts in height $h$, constant with respect to the input length, then $h$ is bounded by an exponential function in the size of $\mathcal{M}$.*

By presenting a suitable family of pushdown automata, we showed that this bound is optimal.

Let us turn our attention to pushdown automata that accept using height which is not constant in the input length, in order to investigate how the pushdown height grows. In particular, we asked if there exists a minimum growth of the pushdown height, with respect to the length of the input, when it is not constant. The answer to this question is already known and it derives from results on Turing machines: the height of the store should grow at least as a double logarithmic function [1]. This lower bound cannot be increased, because a matching upper bound has been recently obtained in [4]. As a consequence of the constructions obtained for automata accepting unary languages, we were able to prove that

▶ **Result 5.6** ([45]). *In the unary case, the height of the store should grow at least as a logarithmic function, and this lower bound cannot be further increased.*

For more details about the results discussed in this section we refer the reader to [45].

## 6 Future Work and Open Problems

In this work we summarized some old and recent results related to the area of descriptional complexity, and, in this regard, we focused on the class of regular languages. To conclude, we discuss possible directions for future research in this field.

First of all, it is worth mentioning that there exist other models characterizing the class of regular languages besides the ones analyzed here. One example are the well-known *regular expressions*, widely discussed in classical textbooks (see, *e.g.* [26]). From regular expressions we can derive a more succinct representation of regular languages, by using *straight-line programs*, namely programs representing directed acyclic graphs, whose internal nodes correspond to the basic regular operations (*i.e.*, union, concatenation, and star). Descriptional complexity of straight-line programs has been analyzed. It has been proved that straight-line programs are polynomially related in size with constant height pushdown automata [11]. Anyways, it would be interesting to deepen the study of descriptional complexity of models *"derived"* from regular expressions.

As widely discussed, the question posed by Sakoda and Sipser in 1978 about the elimination of nondeterminism from finite automata using the two-way motion is still open. We plan to continue the investigation on this question by considering models that have the same computational power as finite automata and by studying the relations in sizes between these nondeterministic devices and deterministic machines. For example, as remarked by Pighizzini in a recent survey, at the moment, direct simulations of 1-limited automata by *deterministic* 1-limited automata and by two-way deterministic automata are not known [41]. It could be interesting to know if, when simulating unary and non-unary 1-limited automata by two-way (instead of one-way) deterministic finite automata, the cost reduces from a double exponential to a simple exponential.

It could be also interesting to study "relaxed" versions of the problem of Sakoda and Sipser, in which the simulating machine is a deterministic 1-limited automaton (*i.e.*, a deterministic two-way automaton with the capability of rewriting the contents of tape cells during the first visit).

Moreover, following the research line started in [10], one could deepen the investigation on the Sakoda and Sipser problem in case of simulated devices that perform a limited use of nondeterminism. In this regard, it is possible to consider several restrictions like, for example, on the number of nondeterministic choices along the computation (see, *e.g.* [25]), or on the number of total, or accepting, computations (also known as *degree of ambiguity* [20, 31, 35]).

## References


1 Maris Alberts. Space complexity of alternating Turing machines. In *Fundamentals of Computation Theory (FCT) '85*, volume 199 of *Lecture Notes in Computer Science*, pages 1–7. Springer, 1985. doi:10.1007/BFb0028785.

2 Marcella Anselmo, Dora Giammarresi, and Stefano Varricchio. Finite automata and non-self-embedding grammars. In *Conference on Implementation and Application of Automata (CIAA) 2002*, volume 2608 of *Lecture Notes in Computer Science*, pages 47–56. Springer, 2002. doi:10.1007/3-540-44977-9_4.

3 Zuzana Bednárová, Viliam Geffert, Carlo Mereghetti, and Beatrice Palano. Removing nondeterminism in constant height pushdown automata. *Information and Computation*, 237:257–267, 2014. doi:10.1016/j.ic.2014.03.002.

4 Zuzana Bednárová, Viliam Geffert, Klaus Reinhardt, and Abuzer Yakaryilmaz. New results on the minimum amount of useful space. *International Journal of Foundations of Computer Science*, 27(2):259–282, 2016. doi:10.1142/S0129054116400098.

5 Piotr Berman and Andrzej Lingas. On the complexity of regular languages in terms of finite automata. Technical Report 304, Polish Academy of Sciences, 1977.

6 Noam Chomsky. A note on phrase structure grammars. *Information and Control*, 2(4):393–395, 1959. doi:10.1016/S0019-9958(59)80017-6.

7 Noam Chomsky. On certain formal properties of grammars. *Information and Control*, 2(2):137–167, 1959. doi:10.1016/S0019-9958(59)90362-6.

8 Noam Chomsky. Context-Free Grammars and Pushdown Storage. Technical Report 65, Massachusetts Institute of Technology, Research Laboratory of Electronics, 1962. URL: https://dspace.mit.edu/bitstream/handle/1721.1/53697/RLE_QPR_065_XVII.pdf.

9 David Gajser. Verifying time complexity of Turing machines. *Theoretical Computer Science*, 600:86–97, 2015. doi:10.1016/j.tcs.2015.07.028.

10 Viliam Geffert, Bruno Guillon, and Giovanni Pighizzini. Two-way automata making choices only at the endmarkers. *Information and Computation*, 239:71–86, 2014. doi:10.1016/j.ic.2014.08.009.

11 Viliam Geffert, Carlo Mereghetti, and Beatrice Palano. More concise representation of regular languages by automata and regular expressions. *Information and Computation*, 208(4):385–394, 2010. doi:10.1016/j.ic.2010.01.002.

12 Viliam Geffert, Carlo Mereghetti, and Giovanni Pighizzini. Converting two-way nondeterministic unary automata into simpler automata. *Theoretical Computer Science*, 295(1–3):189–203, 2003. doi:10.1016/S0304-3975(02)00403-6.

13 Viliam Geffert and Giovanni Pighizzini. Two-way unary automata versus logarithmic space. *Information and Computation*, 209(7):1016–1025, 2011. doi:10.1016/j.ic.2011.03.003.

14 Jonathan Goldstine, Martin Kappes, Chandra M. R. Kintala, Hing Leung, Andreas Malcher, and Detlef Wotschke. Descriptional complexity of machines with limited resources. *Journal of Universal Computer Science*, 8(2):193–234, 2002. doi:10.3217/jucs-008-02-0193.

15 Bruno Guillon, Giovanni Pighizzini, and Luca Prigioniero. Non-self-embedding grammars, constant-height pushdown automata, and limited automata. *International Journal of Foundations of Computer Science*, 31(8):1133–1157, 2020. doi:10.1142/S0129054120420071.

16 Bruno Guillon, Giovanni Pighizzini, Luca Prigioniero, and Daniel Průša. Two-way automata and one-tape machines - read only versus linear time. In *Developments in Language Theory (DLT) 2018*, volume 11088 of *Lecture Notes in Computer Science*, pages 366–378. Springer, 2018. doi:10.1007/978-3-319-98654-8_30.

17 Bruno Guillon, Giovanni Pighizzini, Luca Prigioniero, and Daniel Průša. Converting nondeterministic two-way automata into small deterministic linear-time machines. *CoRR*, abs/2103.05485, 2021. arXiv:2103.05485.

18 Bruno Guillon, Giovanni Pighizzini, Luca Prigioniero, and Daniel Průša. Weight-reducing Turing machines. *CoRR*, abs/2103.05486, 2021. arXiv:2103.05486.

19 Bruno Guillon and Luca Prigioniero. Linear-time limited automata. *Theoretical Computer Science*, 798:95–108, 2019. doi:10.1016/j.tcs.2019.03.037.

20 Yo-Sub Han, Arto Salomaa, and Kai Salomaa. Ambiguity, nondeterminism and state complexity of finite automata. *Acta Cybernetica*, 23(1):141–157, 2017. doi:10.14232/actacyb.23.1.2017.9.

21 Juris Hartmanis. Context-free languages and Turing machine computations. In *Mathematical Aspects of Computer Science*, volume 19 of *Proceedings of Symposia in Applied Mathematics*, pages 42–51. American Mathematical Society, 1967.

22 Fred C. Hennie. One-tape, off-line Turing machine computations. *Information and Control*, 8(6):553–578, 1965. doi:10.1016/S0019-9958(65)90399-2.

23 Thomas N. Hibbard. A generalization of context-free determinism. *Information and Control*, 11(1/2):196–238, 1967. doi:10.1016/S0019-9958(67)90513-X.

24 Markus Holzer and Martin Kutrib. Descriptional complexity — an introductory survey. In *Scientific Applications of Language Methods*, pages 1–58. Imperial College Press, 2010. doi:10.1142/9781848165458_0001.

**25** Markus Holzer and Martin Kutrib. One-time nondeterministic computations. *International Journal of Foundations of Computer Science*, 30(6-7):1069–1089, 2019. doi:10.1142/S012905411940029X.

**26** John E. Hopcroft and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, 1979.

**27** Juraj Hromkovič and Georg Schnitger. Nondeterminism versus determinism for two-way finite automata: Generalizations of Sipser's separation. In *International Colloquium on Automata, Languages, and Programming (ICALP) 2003*, volume 2719 of *Lecture Notes in Computer Science*, pages 439–451. Springer, 2003. doi:10.1007/3-540-45061-0_36.

**28** Christos A. Kapoutsis. Nondeterminism is essential in small two-way finite automata with few reversals. *Information and Computation*, 222:208–227, 2013. doi:10.1016/j.ic.2012.11.001.

**29** Christos A. Kapoutsis. Two-way automata versus logarithmic space. *Theory of Computing Systems*, 55(2):421–447, 2014. doi:10.1007/s00224-013-9465-0.

**30** Christos A. Kapoutsis and Giovanni Pighizzini. Two-way automata characterizations of L/poly versus NL. *Theory of Computing Systems*, 56(4):662–685, 2015. doi:10.1007/s00224-014-9560-x.

**31** Chris Keeler and Kai Salomaa. Nondeterminism growth and state complexity. In *Descriptional Complexity of Formal Systems (DCFS) 2019*, volume 11612 of *Lecture Notes in Computer Science*, pages 210–222. Springer, 2019. doi:10.1007/978-3-030-23247-4_16.

**32** Alica Kelemenová. Complexity of normal form grammars. *Theoretical Computer Science*, 28:299–314, 1984. doi:10.1016/0304-3975(83)90026-9.

**33** Sige-Yuki Kuroda. Classes of languages and linear-bounded automata. *Information and Control*, 7(2):207–223, 1964. doi:10.1016/S0019-9958(64)90120-2.

**34** Martin Kutrib, Giovanni Pighizzini, and Matthias Wendlandt. Descriptional complexity of limited automata. *Information and Computation*, 259(2):259–276, 2018. doi:10.1016/j.ic.2017.09.005.

**35** Hing Leung. Descriptional complexity of nfa of different ambiguity. *International Journal of Foundations of Computer Science*, 16(5):975–984, 2005. doi:10.1142/S0129054105003418.

**36** Carlo Mereghetti and Giovanni Pighizzini. Two-way automata simulations and unary languages. *Journal of Automata, Languages and Combinatorics*, 5(3):287–300, 2000. doi:10.25596/jalc-2000-287.

**37** Albert R. Meyer and Michael J. Fischer. Economy of description by automata, grammars, and formal systems. In *Switching Automata Theory (SwAT) 1971*, pages 188–191. IEEE Computer Society, 1971. doi:10.1109/SWAT.1971.11.

**38** Alexander Okhotin. Non-erasing variants of the Chomsky-Schützenberger theorem. In *Developments in Language Theory (DLT) 2012*, volume 7410 of *Lecture Notes in Computer Science*, pages 121–129. Springer, 2012. doi:10.1007/978-3-642-31653-1_12.

**39** Giovanni Pighizzini. Nondeterministic one-tape off-line Turing machines. *Journal of Automata, Languages and Combinatorics*, 14(1):107–124, 2009. doi:10.25596/jalc-2009-107.

**40** Giovanni Pighizzini. Two-way finite automata: Old and recent results. *Fundamenta Informaticae*, 126(2-3):225–246, 2013. doi:10.3233/FI-2013-879.

**41** Giovanni Pighizzini. Limited automata: Properties, complexity and variants. In *Descriptional Complexity of Formal Systems (DCFS) 2019*, volume 11612 of *Lecture Notes in Computer Science*, pages 57–73. Springer, 2019. doi:10.1007/978-3-030-23247-4_4.

**42** Giovanni Pighizzini and Andrea Pisoni. Limited automata and regular languages. *International Journal of Foundations of Computer Science*, 25(7):897–916, 2014. doi:10.1142/S0129054114400140.

**43** Giovanni Pighizzini and Andrea Pisoni. Limited automata and context-free languages. *Fundamenta Informaticae*, 136(1-2):157–176, 2015. doi:10.3233/FI-2015-1148.

**44** Giovanni Pighizzini and Luca Prigioniero. Limited automata and unary languages. *Information and Computation*, 266:60–74, 2019. doi:10.1016/j.ic.2019.01.002.

**45** Giovanni Pighizzini and Luca Prigioniero. Pushdown automata and constant height: Decidability and bounds. In *Descriptional Complexity of Formal Systems (DCFS) 2019*, volume 11612 of *Lecture Notes in Computer Science*, pages 260–271. Springer, 2019. doi:10.1007/978-3-030-23247-4_20.

**46** Giovanni Pighizzini and Luca Prigioniero. Non-self-embedding grammars and descriptional complexity. *Fundamenta Informaticae*, 180(1-2):103–122, 2021. To appear. doi:10.3233/FI-2021-2036.

**47** Giovanni Pighizzini, Jeffrey O. Shallit, and Ming-wei Wang. Unary context-free grammars and pushdown automata, descriptional complexity and auxiliary space lower bounds. *Journal of Computer and System Sciences*, 65(2):393–414, 2002. doi:10.1006/jcss.2002.1855.

**48** Luca Prigioniero. *Regular Languages: To Finite Automata and Beyond - Succinct Descriptions and Optimal Simulations*. PhD thesis, Università degli Studi di Milano, Dipartimento di Informatica, 2020. doi:10.13130/prigioniero-luca_phd2020-01-31.

**49** Luca Prigioniero. Regular languages: To finite automata and beyond succinct descriptions and optimal simulations. *Bulletin of EATCS*, 131, 2020. URL: http://eatcs.org/beatcs/index.php/beatcs/article/view/619.

**50** Daniel Průša. Weight-reducing Hennie machines and their descriptional complexity. In *Language and Automata Theory and Applications (LATA) 2014*, volume 8370 of *Lecture Notes in Computer Science*, pages 553–564, 2014. doi:10.1007/978-3-319-04921-2_45.

**51** Michael O. Rabin and Dana Scott. Finite automata and their decision problems. *IBM Journal of Research and Development*, 3(2):pp. 114–125, 1959. doi:10.1147/rd.32.0114.

**52** William J. Sakoda and Michael Sipser. Nondeterminism and the size of two way finite automata. In *Symposium on Theory of Computing (SToC) 1978*, pages 275–286. ACM, 1978. doi:10.1145/800133.804357.

**53** John C. Shepherdson. The reduction of two-way automata to one-way automata. *IBM Journal of Research and Development*, 3(2):198–200, 1959. doi:10.1147/rd.32.0198.

**54** Michael Sipser. Halting space-bounded computations. *Theoretical Computer Science*, 10(3):335–338, 1980. doi:10.1016/0304-3975(80)90053-5.

**55** Kohtaro Tadaki, Tomoyuki Yamakami, and Jack C. H. Lin. Theory of one-tape linear-time Turing machines. *Theoretical Computer Science*, 411(1):22–43, 2010. doi:10.1016/j.tcs.2009.08.031.

**56** Klaus W. Wagner and Gerd Wechsung. *Computational Complexity*. D. Reidel Publishing Company, Dordrecht, 1986.

# Reversible Cellular Automata in Presence of Noise Rapidly Forget Everything

**Siamak Taati**
Department of Mathematics, American University of Beirut, Lebanon

## Abstract

We consider reversible and surjective cellular automata perturbed with noise. We show that, in the presence of positive additive noise, the cellular automaton forgets all the information regarding its initial configuration exponentially fast. In particular, the state of a finite collection of cells with diameter $n$ becomes indistinguishable from pure noise after $O(\log n)$ time steps. This highlights the seemingly unavoidable need for irreversibility in order to perform scalable reliable computation in the presence of noise.

**2012 ACM Subject Classification** Hardware → Reversible logic; Hardware → Fault tolerance; Theory of computation → Parallel computing models; Mathematics of computing → Stochastic processes; Mathematics of computing → Information theory

**Keywords and phrases** Reversible cellular automata, surjective cellular automata, noise, probabilistic cellular automata, ergodicity, entropy, reversible computing, reliable computing, fault tolerance

**Digital Object Identifier** 10.4230/OASIcs.AUTOMATA.2021.3

**Category** Invited Talk

# 1 Introduction

## 1.1 Background

A major challenge regarding the physical implementation of computation is the inevitability of transient errors due to noise. The difficulty is that, even if each component of the system is built to be highly accurate and has a very small probability of error, in a lengthy computation, occasional errors are bound to occur. Such errors may then propagate and entirely corrupt the computation. The problem of how to perform computation reliably in the presence of noise (via suitable error-correcting mechanisms) goes back to von Neumann and has been studied since [36, 12].

At the nanoscopic scale, the issue of thermal noise becomes even more pressing. Not only are the nano-scale components more sensitive to any sort of fluctuations, but also the heat generated by the computational process has little time and space to escape the system, and thus leads to an increase in thermal noise. Landauer argued that the heat generated by a computational process is associated to its logical irreversibility, and identified a theoretical lower bound for the amount of heat dissipated in the process of erasing a bit of information [18]. Bennett, Fredkin and Toffoli showed that, at least in theory, any computation can be efficiently simulated by a logically reversible one, hence requiring virtually no dissipation [2, 3, 9, 4]. The output of the reversible computation will consist of the intended output as well as some extra information that allows one to trace the computation backwards. If need be, this extra information can be erased away from the computer core, hence avoiding the accumulation of heat.

By the virtue of their "physics-like" features, cellular automata (CA) have been a popular mathematical model for studying the physical aspects of computation. The notion of reversibility has a natural formulation in the setting of CA, and reversible CA have been widely studied, not only as models of reversible computers, but also as models of physical

27th IFIP WG 1.5 International Workshop on Cellular Automata and Discrete Complex Systems (AUTOMATA 2021).
Editors: Alonso Castillo-Ramirez, Pierre Guillon, and Kévin Perrot; Article No. 3; pp. 3:1–3:15

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

processes and from other points of view [30, 35, 22, 31, 26, 32, 16, 7, 25, 29, 28, 17]. The reliability of computation in the presence of noise is also studied in the setting of cellular automata [33, 10, 13, 5, 6, 11, 23, 14, 20].

Although logical reversibility solves the issue of heat generation in a computational process, it leads to another difficulty. Even if it does not dissipate heat itself, the process is still exposed to external noise. This external noise can potentially be reduced with proper insulation but can never be eliminated altogether. The logical reversibility of the process entails that the noise entering the system is not dissipated and hence accumulates inside the system. This means that, unless one finds a clever workaround, the state of the system will eventually be overcome by noise [3].

For a model of noisy reversible circuits, Aharonov, Ben-Or, Impagliazzo and Nisan [1] identified the limitation imposed by the accumulation of noise. They considered reversible circuits in which errors occur on wires at each "time unit", and proved that the output of such a circuit is indistinguishable from pure noise unless the size of the circuit is exponential in its depth. Conversely, they showed that every classical Boolean circuit with size $s$ and depth $d$ can be simulated by a noisy reversible circuit with size $O\big(s \times 2^{O(d)}\big)$ and depth $O(d)$. In particular, polynomial-size noisy reversible circuits have the power of the complexity class $\mathbf{NC}^1$. They also proved a similar (though not as sharp) result concerning noisy quantum circuits.

The result we present here formulates a similar limitation imposed by the accumulation of noise in the setting of cellular automata. We show that a reversible CA subject to positive additive noise forgets its initial configuration exponentially fast, in the sense that the state of any finite collection of its cells with diameter $n$ becomes indistinguishable from pure noise after $O(\log n)$ number of time steps. It remains open whether any meaningful computation can be done reliably with such a limitation.

Mathematically, the forgetfulness of a reversible CA subject to noise corresponds to the exponential ergodicity of the resulting probabilistic CA. This means that the distribution of the process converges exponentially fast to a unique invariant measure, which in this case is the uniform Bernoulli measure (i.e., the distribution of a configuration chosen by independent coin flips). This result improves upon an earlier partial ergodicity result [20], which was limited to shift-invariant initial measures, and in which the rate of convergence was not identified. The exponential ergodicity of noisy reversible CA is a special case of a more general ergodicity result concerning positive-rate probabilistic CA with Bernoulli invariant measures and their asynchronous counterparts [21].

The structure of the paper is as follows. In Section 1.2, we introduce the setting, and in Section 1.3 we state the main result. The proof of the main result, which is based on entropy, appears in Section 2. Section 3 is dedicated to the interpretation of the result regarding the rapid information loss of reversible computers in the presence of noise. The paper is concluded with some discussions in Section 4.

## 1.2 Setting

### General notation

We will use the notation $\mathbb{N} \triangleq \{0, 1, 2, \ldots\}$ and $\mathbb{Z}^+ \triangleq \{1, 2, 3, \ldots\}$. We will write $[\![n, m]\!]$ to denote the integer interval $\{n, n+1, \ldots, m\}$. We use the notation $x_A$ for the restriction of a function $x$ to a subset $A$ of its domain. We write $Z \sim q$ to indicate that $Z$ is a random variable with distribution $q$. The total variation distance between two probability distributions $p$ and

$q$ on a finite set $A$ is

$$\|q-p\|_{\mathrm{TV}} \triangleq \sup_{E\subseteq A} |q(E)-p(E)| = \frac{1}{2}\sum_{a\in A}|q(a)-p(a)| \ .$$

Throughout this article, $\log(\cdot)$ stands for the natural logarithm.

### Cellular automata

Cellular automata (CA) are abstract models of massively parallel computation. A *configuration* of the model is an assignment of symbols from a finite alphabet $\Sigma$ to every site of the lattice $\mathbb{Z}^d$ (for $d=1,2,\ldots$). The sites of the lattice are called *cells* and the symbol on each cell is referred to as its *state*. At each step of the computation, the states of all cells are simultaneously updated according to a *local rule*. The local rule takes into account the current state of the cell to be updated as well as its neighbours. More specifically, the local rule is a function $f:\Sigma^{\mathcal{N}}\to\Sigma$, where $\mathcal{N}\subseteq\mathbb{Z}^d$ is a finite set indicating the relative positions of the *neighbours* of each cell, possibly including the cell itself. A configuration $x\in\Sigma^{\mathbb{Z}^d}$ is updated to a configuration $Fx\in\Sigma^{\mathbb{Z}^d}$, where $(Fx)_i \triangleq f\big((x_{i+a})_{a\in\mathcal{N}}\big)$ for each cell $i\in\mathbb{Z}^d$. We refer to $F$ as the *global map* of the CA. The computation thus consists in iterating the global map $F$ on an initial configuration. We identify a CA with its global map $F$ and speak of the CA $F$.

### Surjectivity, injectivity, and reversibility

A CA is said to be *surjective* (resp., *injective*, *bijective*) if its global map is surjective (resp., injective, bijective). It is well-known that every injective CA is also surjective, and hence bijective. In fact, the Garden-of-Eden theorem states that surjectivity is equivalent to pre-injectivity [24, 27]. A CA $F$ is said to be *pre-injective* if whenever two distinct configurations $x$ and $y$ agree on all but finitely many cells, their images $Fx$ and $Fy$ are distinct.

A CA $F$ is said to be *reversible* if $F$ is an invertible map and $F^{-1}$ is itself a CA. It follows from a topological argument that every bijective CA is automatically reversible [15]. Thus, injectivity, bijectivity and reversibility are equivalent conditions.

### Noise

In this paper, we are concerned with the effect of transient noise on the computation carried out by a CA. In the presence of noise, random errors might occur during the updates of the cells.

We restrict ourselves to a specific model of noise, namely *additive noise*, although the results of the current paper remain true with the somewhat more general model of *permutation noise* [20]. We (arbitrarily) identify $\Sigma$ with an Abelian group $(\Sigma,+)$. Subject to an additive noise with *noise distribution* $q$, a symbol $a$ is replaced with $a+Z$, where $Z$ is a random variable with distribution $q$. We will assume that the noise distribution $q$ is strictly positive. In particular, $\mathbb{P}(a+Z=b)=q(b-a)>0$ for every $a,b\in\Sigma$.

At each time step of the computation, the state of each cell is first updated according to the local rule of the CA and is then perturbed with positive additive noise. The noise variables at different cells and different time steps are all assumed to be independent.

More specifically, the noise is described by a family $(Z_i^t)_{i\in\mathbb{Z}^d,t\in\mathbb{Z}^+}$ of independent random variables with distribution $q$. The trajectory of the noisy computation starting from a configuration $x$ is given by a sequence of random configurations $(X^t)_{t\in\mathbb{N}}$, where $X^0\triangleq x$, and

$$X_i^t \triangleq f\big((X_{i+a}^{t-1})_{a\in\mathcal{N}}\big) + Z_i^t$$

for every time step $t>0$ and every cell $i\in\mathbb{Z}^d$.

## Probabilistic CA

The noisy computation can be described by a probabilistic CA. In a *probabilistic* CA (PCA), the local rule is probabilistic and the updates at different cells and different time steps are performed independently. More specifically, the local rule is given by a stochastic matrix $\varphi : \Sigma^{\mathcal{N}} \times \Sigma \to [0,1]$ (hence, $\sum_{b\in\Sigma} \varphi(u,b) = 1$ for each $u \in \Sigma^{\mathcal{N}}$). A configuration $x$ is updated to a random configuration $Y$ with distribution

$$\mathbb{P}(Y_A = y_A) = \prod_{i\in A} \varphi\big((x_{i+a})_{a\in\mathcal{N}}, y_i\big)$$

for every finite set $A \subseteq \mathbb{Z}^d$. The role of the global map in the deterministic case is played by a (global) transition kernel $\Phi$ where $\Phi(x,\cdot)$ indicates the distribution of $Y$. (See [34] or [20] for more details.) The trajectory of a PCA $\Phi$ is a Markov process with transition kernel $\Phi$, that is, a sequence $(X^t)_{t\in\mathbb{N}}$ of random configurations such that

- **(i)** Given $X^t$, the configuration $X^{t+1}$ is independent of the configurations $X^0, X^1, \ldots, X^{t-1}$.
- **(ii)** Given $X^t$, the distribution of $X^{t+1}$ is given by $\Phi(X^t,\cdot)$.

In the case of a CA $F$ with additive noise, the local rule of the resulting PCA is given by

$$\varphi(u,b) \triangleq q\big(b - f(u)\big)\ ,$$

where $f$ is the local rule of $F$ and $q$ is the noise distribution.

## Ergodicity

We say that a probability measure $\lambda$ on $\Sigma^{\mathbb{Z}^d}$ is *invariant* under a PCA $\Phi$ if $X^{t+1} \sim \lambda$ whenever $X^t \sim \lambda$. We say that $\Phi$ is *ergodic* if it has a unique invariant measure $\lambda$ and furthermore, for any (possibly random) starting configuration $X^0$, the distribution of $X^t$ converges weakly to $\lambda$. This means that for every finite set $A \subseteq \mathbb{Z}^d$ and every $u \in \Sigma^A$,

$$\mathbb{P}(X^t_A = u) \to \lambda\big(\{\hat{x} : \hat{x}_A = u\}\big) \qquad \text{as } t \to \infty.$$

We can interpret the ergodicity of a PCA $\Phi$ as $\Phi$ "forgetting" its initial configuration. However, the convergence (and hence the process of forgetting the initial configuration) can potentially be slow.

Among the PCA that are ergodic, it is quite common that the convergence towards the unique invariant measure is exponentially fast, in the sense that, for every finite set $A \subseteq \mathbb{Z}^d$,

$$\big\|\mathbb{P}(X^t_A \in \cdot\,) - \lambda\big(\{\hat{x} : \hat{x}_A \in \cdot\,\}\big)\big\|_{\mathrm{TV}} \le \alpha_A \mathrm{e}^{-\beta t}\ ,$$

where $\beta > 0$ is a constant (independent of $A$) and $\alpha_A$ depends on the set $A$ but not on $t$ or the initial configuration $X^0 = x$. (Here, $\mathbb{P}(X^t_A \in \cdot\,)$ stands for the distribution of $X^t_A$ and $\lambda\big(\{\hat{x} : \hat{x}_A \in \cdot\,\}\big)$ for the marginal of $\lambda$ on $A$.) It is the dependence of $\alpha_A$ on the set $A$ that has more relevant information on the speed of convergence. We will discuss this further in Section 3.

In the current paper, the unique invariant measure of the PCA we study will be the *uniform Bernoulli measure*, that is, the distribution of a random configuration in which the states of different cells are chosen uniformly at random from $\Sigma$ and independently from one another.

### 1.3 Statement of the theorem

While the primary interest here is in reversible CA in the presence of noise, our main result remains true for surjective CA. By the *diameter* of a finite set $A \subseteq \mathbb{Z}^d$ we mean the smallest $n \in \mathbb{N}$ such that $A \subseteq u + [\![0, n-1]\!]^d$ for some $u \in \mathbb{Z}^d$.

▶ **Theorem 1** (Surjective CA with additive noise). *Let $F : \Sigma^{\mathbb{Z}^d} \to \Sigma^{\mathbb{Z}^d}$ be a surjective CA. Let $\Phi$ be a PCA describing the perturbation of $F$ with positive additive noise. Then, $\Phi$ is exponentially ergodic with the uniform Bernoulli measure $\lambda$ as the unique invariant measure.*

*More specifically, there exist constants $\alpha, \beta, a, b > 0$ such that if $(X^t)_{t\in\mathbb{N}}$ is a trajectory of $\Phi$ with arbitrary initial configuration, then*

$$\big\|\mathbb{P}(X_A^t \in \cdot\,) - \lambda\big(\{\hat{x} : \hat{x}_A \in \cdot\,\}\big)\big\|_{\mathrm{TV}} \leq \alpha \mathrm{e}^{-\beta t} n^{(d-1)/2}\,,$$

*for every finite set $A \subseteq \mathbb{Z}^d$ with diameter $n$ and every $t \geq a \log n + b$.*

The above theorem completes an earlier partial result in which the uniqueness and convergence (without rate of convergence) was established only among shift-invariant measures [20]. Theorem 1 is a special case (relevant to reversible computing) of a more general result: every probabilistic CA with strictly positive transition probabilities which has a Bernoulli invariant measure is exponentially ergodic [21].

## 2 Accumulation of entropy

Like the result of Aharonov et al. [1], the proof of Theorem 1 is based on entropy. Recall that the (Shannon) *entropy* of a discrete random variable $X$ (measured in *nats*) is defined as

$$H(X) \triangleq -\sum_x \mathbb{P}(X = x) \log \mathbb{P}(X = x)\,.$$

The entropy of $X$ measures the average information content of $X$. If $X$ takes its values within a finite set $\Gamma$, then $H(X) \leq \log|\Gamma|$, with equality if and only if $X$ is uniformly distributed over $\Gamma$. We refer to [8] for information on entropy and its properties.

We follow the approach of the earlier proof of ergodicity modulo shift [20]. Namely, we use the fact that positive additive noise increases the entropy of every finite collection of cells, while a surjective CA does not erase the entropy and only "diffuses" it. In order to achieve complete ergodicity with sharp rate of convergence, we use two new ingredients:

**(a)** An explicit bound on the amount of entropy increase due to noise,

**(b)** A "bootstrap argument" showing that if the rate of entropy increase is high compared to the rate of entropy leakage, then the entropy of each finite set will inevitably accumulate and rise up to its maximum capacity.

That the proof is based on entropy is natural. A reversible CA in the presence of noise can be thought of as a (microscopically reversible) physical system in contact with a heat bath. The entropy increase is therefore a manifestation of the second law of thermodynamics for such systems.

### 2.1 Effect of additive noise on entropy

We set $\overline{h} \triangleq \log|\Sigma|$, so that $\overline{h}$ is the highest possible entropy of a $\Sigma$-valued random variable. Throughout this section, we also let $q : \Sigma \to (0, 1)$ be a strictly positive probability distribution, and set $\kappa \triangleq |\Sigma| \min_{a\in\Sigma} q(a)$. Note that $0 < \kappa \leq 1$. To avoid trivial situations, we assume that $\Sigma$ has at least two elements and that $q$ is not uniform. Hence, $\overline{h} > 0$ and $\kappa < 1$.

▶ **Lemma 2** (Effect of additive noise on entropy). *If $A$ and $N$ are independent $\Sigma$-valued random variables with $N \sim q$, then*

$$H(A+N) \geq \kappa\overline{h} + (1-\kappa)H(A) \ .$$

**Proof.** The distribution $q$ can be decomposed as

$$q = \kappa u + (1-\kappa)\tilde{q}$$

where $u$ is the uniform distribution on $\Sigma$ and $\tilde{q}$ is another distribution on $\Sigma$ given by $\tilde{q}(a) \triangleq \frac{q(a)-\kappa/|\Sigma|}{1-\kappa}$ for $a \in \Sigma$. Thus, without loss of generality (by defining a new probability space if necessary), we can assume that $N = BU + (1-B)\tilde{N}$ where $B \sim \mathsf{Bernoulli}(\kappa)$, $U \sim \mathsf{Uniform}(\Sigma)$ and $\tilde{N} \sim \tilde{q}$, and the variables $A$, $B$, $U$ and $\tilde{N}$ are independent.

Using this representation, we have

$$\begin{aligned}
&H(A+N) \\
&\quad = H(A + BU + (1-B)\tilde{N}) \\
&\quad \geq H(A + BU + (1-B)\tilde{N} \mid B) \\
&\quad = \kappa H(A + \underbrace{BU + (1-B)\tilde{N}}_{=U \text{ when } B=1} \mid B=1) + (1-\kappa)H(A + \underbrace{BU + (1-B)\tilde{N}}_{=\tilde{N} \text{ when } B=0} \mid B=0) \\
&\quad = \kappa\overline{h} + (1-\kappa)H(A+\tilde{N}) \ .
\end{aligned}$$

The claim follows once we recall that $H(A+\tilde{N}) \geq H(A)$ whenever $A$ and $\tilde{N}$ are independent $\Sigma$-valued random variables. Namely, we have

$$\begin{aligned}
H(A+\tilde{N}, \tilde{N}) &= H(\tilde{N}) + \overbrace{H(A+\tilde{N} \mid \tilde{N})}^{=H(A)} \\
H(A+\tilde{N}, \tilde{N}) &= H(A+\tilde{N}) + H(\tilde{N} \mid A+\tilde{N})
\end{aligned}$$

from which we get $H(A+\tilde{N}) = H(A) + I(A+\tilde{N}; \tilde{N})$ where $I(A+\tilde{N}; \tilde{N}) \geq 0$ is the mutual information between $A+\tilde{N}$ and $\tilde{N}$. ◀

The conditional version of the above lemma can be proven similarly, or by reducing it to the unconditional version.

▶ **Lemma 3** (Effect of additive noise on conditional entropy). *If $A$ and $N$ are $\Sigma$-valued random variables and $C$ is another random variable conditioned on which $A$ and $N$ are independent with $N \sim q$, then*

$$H(A+N \mid C) \geq \kappa\overline{h} + (1-\kappa)H(A \mid C) \ .$$

For a collection of random symbols subjected to independent noise, we have the following lemma as a corollary.

▶ **Lemma 4** (Effect of additive noise on joint entropy). *If $\underline{A} \triangleq (A_1, A_2, \ldots, A_n)$ and $\underline{N} \triangleq (N_1, N_2, \ldots, N_n)$ are two independent collections of $\Sigma$-valued random variables and $N_1, N_2, \ldots, N_n$ are i.i.d. with distribution $q$, then*

$$H(\underline{A} + \underline{N}) \geq n\kappa\overline{h} + (1-\kappa)H(\underline{A}) \ .$$

**Proof.** Using the chain rule and the fact that $N_k$'s are independent of one another and independent $A_k$'s, we have

$$H(\underline{A}) = \sum_{k=1}^{n} H\big(A_k \,\big|\, (A_i)_{i<k}\big) \tag{1}$$

and

$$\begin{aligned} H(\underline{A}+\underline{N}) &= \sum_{k=1}^{n} H\big(A_k + N_k \,\big|\, (A_i+N_i)_{i<k}\big) \\ &\geq \sum_{k=1}^{n} H\big(A_k + N_k \,\big|\, (A_i)_{i<k}, (N_i)_{i<k}\big) \\ &= \sum_{k=1}^{n} H\big(A_k + N_k \,\big|\, (A_i)_{i<k}\big)\ . \end{aligned} \tag{2}$$

Applying Lemma 3 to the corresponding terms in (1) and (2) yields the result. ◀

## 2.2 Effect of a surjective CA on entropy

The effect of surjective CA on entropy was clarified in the earlier proof of ergodicity modulo shift [20]. For completeness, we recall the proof. Given a set $J \subseteq \mathbb{Z}^d$ and an integer $r \geq 0$, we denote by $\mathcal{M}^r(J) \triangleq J + [\![-r, r]\!]^d$ the set of all cells that are within distance $r$ from $J$. We also define $\partial\mathcal{M}^r(J) \triangleq \mathcal{M}^r(J) \setminus J$.

▶ **Lemma 5** (Effect of a surjective CA on entropy). *Let $F : \Sigma^{\mathbb{Z}^d} \to \Sigma^{\mathbb{Z}^d}$ be a surjective CA with neighbourhood $\mathcal{N} \subseteq [\![-r, r]\!]$. Then, for every random configuration $X$ and every finite set $J \subseteq \mathbb{Z}^d$ we have*

$$H\big((FX)_J\big) \geq H(X_J) - c(J)$$

*where $c(J) \triangleq \big(\big|\partial\mathcal{M}^{2r}(J)\big| + \big|\partial\mathcal{M}^r(J)\big|\big)\overline{h}$.*

**Proof.** By the Garden-of-Eden theorem, $F$ is pre-injective [24, 27]. From the pre-injectivity of $F$ it follows that for every configuration $x$, the pattern $x_J$ is uniquely determined from the patterns $x_{\partial\mathcal{M}^{2r}(J)}$ and $(Fx)_{\mathcal{M}^r(J)}$. Indeed, suppose that $x$ and $x'$ are two configurations such that $x'_{\partial\mathcal{M}^{2r}(J)} = x_{\partial\mathcal{M}^{2r}(J)}$ and $(Fx')_{\mathcal{M}^r(J)} = (Fx)_{\mathcal{M}^r(J)}$. Let $x''$ be another configuration that agrees with $x'$ on $\mathcal{M}^{2r}(J)$ and with $x$ outside $J$. Then, $x$ and $x''$ agree everywhere except possibly on $J$. On the other hand, $Fx = Fx''$ because

- $Fx''$ and $Fx$ agree outside $\mathcal{M}^r(J)$ because $x''$ and $x$ agree outside $J$,
- $Fx''$ and $Fx'$ agree on $\mathcal{M}^r(J)$ because $x''$ and $x'$ agree on $\mathcal{M}^{2r}(J)$, and $Fx'$ and $Fx$ agree on $\mathcal{M}^r(J)$ by assumption.

The pre-injectivity of $F$ now implies $x'' = x$. It follows that $x$ and $x'$ agree on $J$.

Now, consider a random configuration $X$. Since $X_J$ is a function of $X_{\partial\mathcal{M}^{2r}(J)}$ and $(FX)_{\mathcal{M}^r(J)}$, we have

$$\begin{aligned} H(X_J) &\leq H\big(X_{\partial\mathcal{M}^{2r}(J)}, (FX)_{\mathcal{M}^r(J)}\big) \\ &= H\big((FX)_J\big) + H\big(X_{\partial\mathcal{M}^{2r}(J)}, (FX)_{\partial\mathcal{M}^r(J)} \,\big|\, (FX)_J\big) \\ &\leq H\big((FX)_J\big) + \big(\big|\partial\mathcal{M}^{2r}(J)\big| + \big|\partial\mathcal{M}^r(J)\big|\big)\overline{h}\ , \end{aligned}$$

proving the claim. ◀

## 2.3 Evolution of entropy

Combining the above lemmas we obtain the following proposition.

▶ **Proposition 6** (Evolution of entropy). *Let $\Phi$ be a PCA describing the perturbation of a surjective CA $F : \Sigma^{\mathbb{Z}^d} \to \Sigma^{\mathbb{Z}^d}$ with positive additive noise, and let $q : \Sigma \to (0,1)$ denote the noise distribution. Let $X^0, X^1, \dots$ be a trajectory of $\Phi$ starting from an arbitrary configuration $X^0$. Then, for every finite $J \subseteq \mathbb{Z}^d$ and each $t \geq 0$ we have*

$$H\big(X^t_J\big) \geq [1-(1-\kappa)^t]|J|\overline{h} - \tilde{c}(J)$$

*where $\kappa \triangleq |\Sigma| \min_{a\in\Sigma} q(a)$ and $\tilde{c}(J) \triangleq \big(\frac{1-\kappa}{\kappa}\big)\big(\big|\partial\mathcal{M}^{2r}(J)\big| + \big|\partial\mathcal{M}^{r}(J)\big|\big)\overline{h}$.*

**Proof.** According to Lemma 5, for every $s > 0$ we have

$$H\big((FX^{s-1})_J\big) \geq H(X^{s-1}_J) - c(J)\ .$$

Lemma 4 on the other hand gives

$$H(X^s_J) \geq \kappa|J|\overline{h} + (1-\kappa)H\big((FX^{s-1})_J\big)\ .$$

Combining the two, we find that for $s > 0$,

$$H(X^s_J) \geq (1-\kappa)H(X^{s-1}_J) + \kappa|J|\overline{h} - (1-\kappa)c(J)\ .$$

Multiplying by $(1-\kappa)^{t-s}$, we obtain

$$(1-\kappa)^{t-s}H(X^s_J) \geq (1-\kappa)^{t-s+1}H(X^{s-1}_J) + \kappa(1-\kappa)^{t-s}|J|\overline{h} - (1-\kappa)^{t-s+1}c(J)\ .$$

Summing over $s$ from 1 to $t$, we get

$$\begin{aligned}\sum_{s=1}^{t}(1-\kappa)^{t-s}H(X^s_J) \geq \sum_{s=1}^{t}&(1-\kappa)^{t-s+1}H(X^{s-1}_J)\\ &+ \big[1-(1-\kappa)^t\big]|J|\overline{h} - \big[\overbrace{1-(1-\kappa)^t}^{<1}\big]\frac{1-\kappa}{\kappa}c(J)\end{aligned}$$

which after cancellation of the common terms gives

$$\begin{aligned}H(X^t_J) &\geq (1-\kappa)^t H(X^0_J) + \big[1-(1-\kappa)^t\big]|J|\overline{h} - \tilde{c}(J)\\ &\geq \big[1-(1-\kappa)^t\big]|J|\overline{h} - \tilde{c}(J)\ .\end{aligned}$$

◀

As a corollary, we get the following proposition.

▶ **Proposition 7** (Evolution of entropy). *Let $\Phi$ be a PCA describing the perturbation of a surjective CA $F : \Sigma^{\mathbb{Z}^d} \to \Sigma^{\mathbb{Z}^d}$ with positive additive noise, and let $q : \Sigma \to (0,1)$ denote the noise distribution. There are two constants $a_0, b_0 > 0$ with the following property. If $X^0, X^1, \dots$ is a trajectory of $\Phi$ starting from an arbitrary configuration $X^0$, then for every finite set $J \subseteq \mathbb{Z}^d$, we have*

$$H(X^t_J) \geq |J|\overline{h} - 2\tilde{c}(J) \qquad \text{for all } t \geq a_0 \log\frac{|J|}{\tilde{c}(J)} + b_0$$

*where $\tilde{c}(J) \triangleq \big(\frac{1-\kappa}{\kappa}\big)\big(\big|\partial\mathcal{M}^{2r}(J)\big| + \big|\partial\mathcal{M}^{r}(J)\big|\big)\overline{h}$.*

**Proof.** From Proposition 6, it follows that in order to have $H(X^t_J) \geq |J|\overline{h} - 2\tilde{c}(J)$, it is sufficient that $(1-\kappa)^t|J|\overline{h} \leq \tilde{c}(J)$. We have,

$$\begin{aligned}(1-\kappa)^t|J|\overline{h} \leq \tilde{c}(J) &\iff t\log(1-\kappa) \leq \log\tilde{c}(J) - \log|J| - \log\overline{h}\\ &\iff t \geq \frac{\log|J| - \log\tilde{c}(J) + \log\overline{h}}{-\log(1-\kappa)}\\ &\iff t \geq a_0 \log\frac{|J|}{\tilde{c}(J)} + b_0\end{aligned}$$

where $a_0 \triangleq -1/\log(1-\kappa)$ and $b_0 \triangleq -\log\overline{h}/\log(1-\kappa)$. ◀

The latter proposition can be interpreted as follows. For $n \geq 0$, consider a hypercube $S_n \triangleq [\![0, n-1]\!]^d$ of size $n^d$ in the lattice. Then, $\tilde{c}(S_n) = \Theta(n^{d-1})$ as $n \to \infty$. Thus, according to Proposition 7, irrespective of the distribution of the initial configuration $X^0$, we have $H(X^t_{S_n}) \geq n^d\overline{h} - \Theta(n^{d-1})$ (i.e., $S_n$ lacks no more than $\Theta(n^{d-1})$ nats of entropy at time $t$) as soon as $t \geq \Theta(\log n)$.

## 2.4 A bootstrap lemma

The next step is a "bootstrap argument". The intuitive idea is as follows. The effect of noise on a hypercube $S_n \triangleq [\![0, n-1]\!]^d$ is to accumulate entropy as long as the entropy of $S_n$ is less than its maximum capacity $|S_n|\overline{h}$. A surjective CA on the other hand keeps the entropy of $S_n$ almost preserved except for a leakage of size $O(|\partial S_n|)$ per iteration through the boundary of $S_n$. The "equilibrium" is reached at time $t_n = O(\log n)$, when the rate of accumulation and the maximum rate of leakage roughly match. Now consider a much larger hypercube $S_m$ which contains many disjoint copies of $S_n$. For $S_m$, a similar "equilibrium" is reached at time $t_m = O(\log m)$. However, the entropy leaking from the copies of $S_n$ will not have enough time to reach and escape through the boundary of $S_m$ before time $t_m$, and will hence have to accumulate inside $S_m$. This implies that the entropy missing from each copy of $S_n$ at time $t_m$ must in fact be much less than $\Theta(|\partial S_n|)$.

Let us make this argument precise. Given a random configuration $X$ and finite set $A \subseteq \mathbb{Z}^d$, let us write $\Xi(X_A) \triangleq |A|\overline{h} - H(X_A)$ for the difference between the entropy of $X_A$ and the maximum entropy capacity of $A$. Note that $\Xi(X_A) \geq 0$ with equality if and only if $X_A$ is uniformly distributed over $\Sigma^A$.

▶ **Lemma 8** (Bootstrap lemma). *Let $\Phi$ be a PCA on $\Sigma^{\mathbb{Z}^d}$ with neighbourhood $\mathcal{N} \subseteq [\![-r, r]\!]^d$. Let $\tau, \delta : \mathbb{Z}^+ \to [0, \infty)$ be two functions satisfying the following property:*

- *for every trajectory $X^0, X^1, \dots$ of $\Phi$ and each $n \in \mathbb{Z}^+$, we have $\Xi(X^t_{S_n}) \leq \delta(n)$ for all $t \geq \tau(n)$, irrespective of the distribution of $X^0$.*

*Let $k, m, n, t \in \mathbb{Z}^+$ be such that $m \geq k(n + 2rt)$ and $t \geq \tau(m)$. Then, for every trajectory $X^0, X^1, \dots$ of $\Phi$, we have $\Xi(X^t_{S_n}) \leq \delta(m)/k^d$.*

**Proof.** Observe that we can pack $k^d$ disjoint copies of $\mathcal{M}^{rt}(S_n)$ in $S_m$. Namely, for $w \in S_k$, let $Q_w \triangleq (n + 2rt)w + [\![rt, rt + n - 1]\!]^d$. Then, the sets $\mathcal{M}^{rt}(Q_w)$ (for $w \in S_k$) are disjoint and are all included in $S_m$. Construct a random configuration $Y$ by choosing the patterns $Y_{\mathcal{M}^{rt}(Q_w)}$ (for $w \in S_k$) independently according to the distribution of $X^0_{\mathcal{M}^{rt}(S_n)}$, and assigning arbitrary values to the remaining cells. Consider a trajectory $Y^0, Y^1, \dots$ of $\Phi$ with initial configuration $Y^0 \triangleq Y$. Clearly, the patterns $Y^t_{Q_w}$ (for $w \in S_k$) are independent and have the

same distribution as $X^t_{S_n}$. Therefore, using the chain rule, we have

$$H(Y^t_{S_m}) = \sum_{w\in S_k} H(Y^t_{Q_w}) + H\left(Y^t_{S_m\setminus\bigcup_{w\in S_k} Q_w} \,\middle|\, Y^t_{\bigcup_{w\in S_k} Q_w}\right)$$
$$\le k^d H(X^t_{S_n}) + \left|S_m \setminus \bigcup_{w\in S_k} Q_w\right| \overline{h}\,,$$

which implies

$$\Xi(Y^t_{S_m}) \ge k^d \Xi(X^t_{S_n})\,.$$

Now, since $Y^0, Y^1, \ldots$ is a trajectory of $\Phi$ and $t \ge \tau(m)$, we have $\Xi(Y^t_{S_m}) \le \delta(m)$. It follows that $\Xi(X^t_{S_n}) \le \delta(m)/k^d$, as claimed. ◀

## 2.5 Proof of the theorem

**Proof of Theorem 1.** Let $X^0, X^1, \ldots$ be a trajectory of $\Phi$. For every finite set $A \subseteq \mathbb{Z}^d$, we show that $\Xi(X^t_A) = |A|\overline{h} - H(X^t_A) \to 0$ exponentially fast as $t \to \infty$. This would imply that the distribution of $X^t$ converges weakly to the uniform Bernoulli measure $\lambda$. We then translate the bound on $\Xi(X^t_A)$ to a bound on total variation distance.

Let $r \in \mathbb{N}$ be such that the neighbourhood of $F$ (and hence also of $\Phi$) is included in $[\![-r, r]\!]^d$. Let $\tilde{c}(J)$ and $a_0, b_0 > 0$ be as in Proposition 7, and note that $\tilde{c}(S_n) = \Theta(n^{d-1})$ as $n \to \infty$. Choose constants $a_1, b_1, c_1 > 0$ such that $\tau(n) \triangleq a_1 \log n + b_1 \ge a_0 \log \frac{|S_n|}{\tilde{c}(S_n)} + b_0$ and $\delta(n) \triangleq c_1 n^{d-1} \ge 2\tilde{c}(S_n)$ for every $n \in \mathbb{Z}^+$. Then, by Proposition 7, for each $n \in \mathbb{Z}^+$ we have $\Xi(X^t_{S_n}) \le \delta(n)$ whenever $t \ge \tau(n)$, hence the hypothesis of Lemma 8 is satisfied.

Suppose that $A \subseteq \mathbb{Z}^d$ is a finite set of cells with diameter $n$. This means that $A \subseteq u + S_n$ for some $u \in \mathbb{Z}^d$. For $t \ge 0$, define $m_t \triangleq k_t(n + 2rt)$, where $k_t \in \mathbb{Z}^+$ is to be determined. Then, according to Lemma 8,

$$\Xi(X^t_A) \le \frac{\delta(m_t)}{k_t^d} = \frac{c_1 k_t^{d-1}(n+2rt)^{d-1}}{k_t^d} = c_1 k_t^{-1}(n+2rt)^{d-1}\,,$$

provided that

$$t \ge \tau(m_t) = a_1 \log k_t + a_1 \log(n + 2rt) + b_1\,. \tag{3}$$

Now, pick $\beta_1$ such that $0 < \beta_1 < 1/a_1$, and set $k_t \triangleq \lfloor \mathrm{e}^{\beta_1 t} \rfloor$. Observe that with this choice, condition (3) is satisfied for all sufficiently large $t$. In particular, we can find constants $a, b > 0$ such that (3) holds whenever $t \ge a \log n + b$. It follows that, for a suitable constant $c_2 > 0$, $\Xi(X^t_A) \le c_2 \mathrm{e}^{-\beta_1 t}(n + 2rt)^{d-1}$ for all $t \ge a \log n + b$. (Here, we need a new constant $c_2$ instead of $c_1$ in order to compensate for replacing $\lfloor \mathrm{e}^{\beta_1 t} \rfloor$ with $\mathrm{e}^{\beta_1 t}$.) In particular, $\Xi(X^t_A) \to 0$ exponentially fast as $t \to \infty$.

Next, let $\mu^t$ denote the distribution of $X^t$, and denote by $\mu^t_A$ and $\lambda_A$ the marginals of $\mu^t$ and $\lambda$ on $A$. Observe that $\Xi(X^t_A)$ is the same as $D(\mu^t_A \| \lambda_A)$, the Kullback–Leibler divergence of $\mu^t_A$ relative to $\lambda_A$. According to Pinsker's inequality [8, Lemma 11.6.1], we have

$$\|\mu^t_A - \lambda_A\|_{\mathrm{TV}} \le \sqrt{\frac{1}{2} D(\mu^t_A \| \lambda_A)} = \sqrt{\frac{1}{2}\Xi(X^t_A)}\,.$$

We conclude that

$$\|\mu^t_A - \lambda_A\|_{\mathrm{TV}} \le \sqrt{c_2/2}\, \mathrm{e}^{-(\beta_1/2)t}(n + 2rt)^{(d-1)/2}$$

for every finite $A \subseteq \mathbb{Z}^d$ with diameter $n$ and all $t \ge a \log n + b$. The claim follows by choosing $\beta > 0$ slightly smaller than $\beta_1/2$ and $\alpha > 0$ sufficiently large. ◀

# 3 Rapid information loss

## 3.1 Reversible CA on an infinite lattice

Theorem 1 shows that for a surjective CA subject to positive additive noise, mixing occurs quite fast, in the sense that it takes only $O(\log n)$ steps before the marginal on each hypercube of size $n^d$ is within $\varepsilon$-distance from its stationary value.

To make this precise, let us use the notation

$$\|\mu - \nu\|_A \triangleq \|\mu_A - \nu_A\|_{\mathrm{TV}} = \frac{1}{2} \sum_{u \in \Sigma^A} |\mu([u]) - \nu([u])|$$

for the total variation distance between the marginals of two measures $\mu$ and $\nu$ on $A \Subset \mathbb{Z}^d$. Let

$$d_A(t) \triangleq \sup_{x \in \Sigma^{\mathbb{Z}^d}} \|\Phi^t(x, \cdot) - \lambda\|_A$$

be the maximum distance from stationarity of the marginal on $A$ at time $t$. Note that $d_A(t)$ is non-increasing with $t$. Given a finite set $A \subseteq \mathbb{Z}^d$ and $\varepsilon > 0$, we let

$$t_{\mathsf{mix}}(A, \varepsilon) \triangleq \inf\{t : d_A(t) \leq \varepsilon\} .$$

We call $t_{\mathsf{mix}}(A, \varepsilon)$ the *mixing time* of set $A$ at accuracy level $\varepsilon$ (cf. [19]).

As before, we let $S_n \triangleq [\![0, n-1]\!]^d$ be a hypercube of size $n^d$ in the lattice.

▶ **Corollary 9** (Mixing time of surjective CA with additive noise). *Let $\Phi$ be a PCA describing the perturbation of a surjective CA $F : \Sigma^{\mathbb{Z}^d} \to \Sigma^{\mathbb{Z}^d}$ with positive additive noise. For every $\varepsilon > 0$, we have $t_{\mathsf{mix}}(S_n, \varepsilon) = O(\log n)$ as $n \to \infty$.*

**Proof.** According to Theorem 1,

$$d_{S_n}(t) \leq \alpha \mathrm{e}^{-\beta t} n^{(d-1)/2}$$

for every $n \in \mathbb{Z}^+$ and $t \geq a \log n + b$. Therefore, $d_{S_n}(t) \leq \varepsilon$ as soon as

$$t \geq \max\left\{a \log n + b, \frac{d-1}{2\beta} \log n + \frac{1}{\beta}(\log \alpha - \log \varepsilon)\right\} ,$$

which means $t_{\mathsf{mix}}(S_n, \varepsilon) = O(\log n)$. ◀

In other words, for every $n > 0$ and any accuracy $\varepsilon > 0$, the distribution of the pattern on $S_n$ becomes $\varepsilon$-indistinguishable from the uniform distribution after $O(\log n)$ time steps.

## 3.2 Reversible parallel computers with finite space

In the proof of Theorem 1, the bootstrap lemma was needed to handle the potential diffusion of entropy on the infinite lattice. For a reversible parallel computer with finite space, a simpler entropy argument can be used to show that, in the presence of positive additive noise, the state of the system becomes indistinguishable from pure noise in logarithmic number of steps. This is a reformulation of Theorem 2 of Aharonov et al. [1] in a slightly more general setup.

We consider a model of parallel reversible computation with finite space in which every piece of data is (reversibly) processed at each time step, and is hence exposed to noise. More specifically, let $A$ be a finite set and $\Sigma$ a finite alphabet. Let $\mathscr{F}$ be a family of bijective maps

$F : \Sigma^A \to \Sigma^A$. Consider a computational process whose data is an element of $\Sigma^A$, and in which, at every step, an arbitrary map from $\mathscr{F}$ is applied to the data, and the result is then subjected to positive additive noise. For instance, a reversible logic circuit with $t$ layers, $n$ nodes per layer, and wires between consecutive layers only, in which every node is subject to positive noise fits in this setting.

The above process can be described by a time-inhomogeneous finite-state Markov chain which mixes rapidly. To make this precise, let us review some terminology and notation regarding finite-state Markov chains [19]. Let $\Phi$ be an ergodic (possibly time-inhomogeneous) Markov chain with finite state space $\mathcal{X}$ and unique stationary distribution $\pi$. We write $\Phi^{s\to t}$ for the transition matrix from time $s$ to time $t$. Let

$$d_\Phi(t) \triangleq \sup_{x\in\mathcal{X}} \|\Phi^{0\to t}(x,\cdot) - \pi\|_{\mathrm{TV}}$$

denote the maximum distance from stationarity of the distribution at time $t$. Since $\Phi$ is ergodic, $d(t) \to 0$ monotonically as $t \to \infty$. The *mixing time* at accuracy $\varepsilon > 0$ is defined as

$$t_{\mathsf{mix}}(\Phi,\varepsilon) \triangleq \inf\{t : d_\Phi(t) \le \varepsilon\} .$$

▶ **Theorem 10** (Rapid mixing of reversible computer with noisy components)**.** *Let $A$ be a finite set and $\Sigma$ a finite alphabet. Let $\mathscr{F}$ be a family of bijective maps $F : \Sigma^A \to \Sigma^A$. Let $\Phi$ be a time-inhomogeneous Markov chain on $\Sigma^A$ in which, at each time step, first an (arbitrary) element of $\mathscr{F}$ is applied to the current state, and then the state is subjected to positive additive noise with fixed noise distribution. Then, $\Phi$ is ergodic with the uniform distribution on $\Sigma^A$ as the stationary distribution. Furthermore, for every $\varepsilon > 0$, $t_{\mathsf{mix}}(\Phi,\varepsilon) = O(\log|A|)$ as $|A| \to \infty$.*

**Proof.** Let $X^t$ denote the state of the Markov chain at time $t$. Let $\lambda$ denote the uniform distribution on $\Sigma^A$. Let $F_t \in \mathscr{F}$ be the map applied at time $t$, and let $q$ denote the noise distribution. Thus, $X^t$ is obtained from $X^{t-1}$ by applying additive noise with distribution $q$ to $F_t(X^{t-1})$. Let $\kappa \triangleq |\Sigma| \min_{a\in\Sigma} q(a)$.

The bijective maps $F_t$ do not change the entropy, hence according to Lemma 4,

$$\begin{aligned} H(X^t) &\ge |A|\kappa\overline{h} + (1-\kappa)H\big(F_t(X^{t-1})\big) \\ &= |A|\kappa\overline{h} + (1-\kappa)H(X^{t-1}) \end{aligned}$$

for each $t \ge 1$. Therefore, setting $\Xi(X^t) \triangleq |A|\overline{h} - H(X^t)$, we have

$$\Xi(X^t) \le (1-\kappa)^t \Xi(X^0) \le (1-\kappa)^t |A|\overline{h}$$

for every $t \ge 0$. This shows that the Markov chain is ergodic with $\lambda$ as the unique stationary distribution.

Now, recall that $\Xi(X^t)$ is the same as the Kullback–Leibler divergence $D(\mu^t \,\|\, \lambda)$, where $\mu^t$ is the distribution of the Markov chain at time $t$. Therefore, Pinsker's inequality [8, Lemma 11.6.1] gives

$$\|\mu^t - \lambda\|_{\mathrm{TV}} \le \sqrt{\frac{1}{2} D(\mu^t \,\|\, \lambda)} = \sqrt{\frac{1}{2}\Xi(X_A^t)} \le \sqrt{\overline{h}/2}\,|A|^{1/2}(1-\kappa)^{t/2} .$$

Since $X^0$ is arbitrary, we get

$$d_\Phi(t) \le \sqrt{\overline{h}/2}\,|A|^{1/2}(1-\kappa)^{t/2} ,$$

from which it follows that $t_{\mathsf{mix}}(\Phi,\varepsilon) = O(\log|A|)$ as $|A| \to \infty$. ◀

## 4 Discussion

### Is there anything we can do with a noisy reversible CA?

Since an input of size $n$ is lost in $O(\log n)$ steps, no cell on the lattice will have time to "sense" all the input. Thus, it seems unlikely that one can do any meaningful computation with a noisy reversible CA in a scalable fashion.

Theorem 1 of Aharoni et al. [1] states that, with exponential redundancy, any logic circuit can be simulated by a reliable reversible circuit. In order to simulate a computation with $d$ steps (i.e., a circuit of depth $d$), every input bit is provided in $3^d$ separate copies. The $i$-th step of the computation is performed $3^{d-i+1}$ times in parallel on separate copies, and majority vote on groups of 3 is used to pass the results of the $i$-th step to the next step. Such an error-correcting mechanism cannot be implemented in the setting of cellular automata on $\mathbb{Z}^d$, because the dependence graph of each cell grows at most polynomially.

### Reversible serial computers

Theorem 10 describes the loss of information in a parallel model of a reversible computer in which every bit of data is updated at every step of the computation and is therefore subjected to noise. In a serial computer (such as a Turing machine) on the other hand, only a bounded portion of data is operated on at each step, and only that portion is affected by significant noise. We can there wonder about the possibility of having a reversible serial computer that is capable of performing a significantly long computation reliably in the presence of noise.

### How much irreversibility is needed to do reliable computation with a CA?

Theorem 1 suggests that, in order to perform reliable computation (with a CA-like computer), some degree of irreversibility is unavoidable. Can we quantify the degree of irreversibility needed to perform reliable computation at a given noise level?

## References


1 D. Aharonov, M. Ben-Or, R. Impagliazzo, and N. Nisan. Limitations of noisy reversible computation, 1996. arXiv:quant-ph/9611028.

2 C. H. Bennett. Logical reversibility of computation. *IBM Journal of Research and Development*, 17(6):525–532, 1973. doi:10.1147/rd.176.0525.

3 C. H. Bennett. The thermodynamics of computation—a review. *International Journal of Theoretical Physics*, 21(12):905–940, 1982. doi:10.1007/BF02084158.

4 C. H. Bennett. Time/space trade-offs for reversible computation. *SIAM Journal on Computing*, 18(4):766–776, 1989. doi:10.1137/0218053.

5 P. Berman and J. Simon. Investigations of fault-tolerant networks of computers. In *STOC '88: Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 66–77. ACM, 1988. doi:10.1145/62212.62219.

6 M. Bramson and C. Neuhauser. Survival of one-dimensional cellular automata under random perturbations. *The Annals of Probability*, 22(1):244–263, 1994. doi:10.1214/aop/1176988858.

7 B. Chopard and M. Droz. *Cellular Automata Modeling of Physical Systems*. Cambridge University Press, 1998. doi:10.1017/CBO9780511549755.

8 T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley, second edition, 2006. doi:10.1002/047174882X.

9 E. Fredkin and T. Toffoli. Conservative logic. *International Journal of Theoretical Physics*, 21(3/4):219–253, 1982. doi:10.1007/BF01857727.

10 P. Gács. Reliable computation with cellular automata. *Journal of Computer and System Sciences*, 32(1):15–78, 1986. doi:10.1016/0022-0000(86)90002-4.

11 P. Gács. Reliable cellular automata with self-organization. *Journal of Statistical Physics*, 103(1–2):45–267, 2001. doi:10.1023/A:1004823720305.

12 P. Gács. Reliable computation, 2005. URL: https://www.cs.bu.edu/fac/gacs/.

13 P. Gács and J. Reif. A simple three-dimensional real-time reliable cellular array. *Journal of Computer and System Sciences*, 36(2):125–147, 1988. doi:10.1016/0022-0000(88)90024-4.

14 P. Gács and I. Törmä. Stable multi-level monotonic eroders, 2018. arXiv:1809.09503.

15 G. Hedlund. Endomorphisms and automorphisms of shift dynamical systems. *Mathematical Systems Theory*, 3:320–375, 1969. doi:10.1007/BF01691062.

16 J. Kari. Reversibility and surjectivity problems of cellular automata. *Journal of Computer and System Sciences*, 48(1):149–182, 1994. doi:10.1016/S0022-0000(05)80025-X.

17 J. Kari. Reversible cellular automata: From fundamental classical results to recent developments. *New Generation Computing*, 36:145–172, 2018. doi:10.1007/s00354-018-0034-6.

18 R. Landauer. Irreversibility and heat generation in the computing process. *IBM Journal of Research and Development*, 5(3):183–191, 1961. doi:10.1147/rd.53.0183.

19 D. A. Levin, Y. Peres, and E. L. Wilmer. *Markov Chains and Mixing Times*. American Mathematical Society, 2009.

20 I. Marcovici, M. Sablik, and S. Taati. Ergodicity of some classes of cellular automata subject to noise. *Electronic Journal of Probability*, 24(41), 2019. doi:10.1214/19-EJP297.

21 I. Marcovici and S. Taati, In preparation.

22 N. Margolus. Physics-like models of computation. *Physica D: Nonlinear Phenomena*, 10(1–2):81–95, 1984. doi:10.1016/0167-2789(84)90252-5.

23 M. McCann and N. Pippenger. Fault tolerance in cellular automata at high fault rates. *Journal of Computer and System Sciences*, 74(5):910–918, 2008. doi:10.1016/j.jcss.2008.02.003.

24 E. F. Moore. Machine models of self-reproduction. In *Mathematical Problems in the Biological Sciences*, volume 14 of *Proceedings of Symposia in Applied Mathematics*, pages 17–33. American Mathematical Society, 1962.

25 K. Morita. Reversible computing and cellular automata—a survey. *Theoretical Computer Science*, 395(1):101–131, 2008. doi:10.1016/j.tcs.2008.01.041.

26 K. Morita and M. Harao. Computation universality of one-dimensional reversible (injective) cellular automata. *The Transactions of the IEICE*, E72(6):758–762, 1989.

27 J. Myhill. The converse of Moore's Garden-of-Eden theorem. *Proceedings of the American Mathematical Society*, 14(4):685–686, 1963. doi:10.2307/2034301.

28 V. Salo and I. Törmä. A one-dimensional physically universal cellular automaton. In J. Kari, F. Manea, and I. Petre, editors, *CiE 2017: Unveiling Dynamics and Complexity*, volume 10307 of *LNCS*, pages 375–386. Springer, 2017. doi:10.1007/978-3-319-58741-7_35.

29 L. Schaeffer. A physically universal cellular automaton. In *ITCS '15: Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science*, pages 237–246. ACM, 2015. doi:10.1145/2688073.2688107.

30 T. Toffoli. Computation and construction universality of reversible cellular automata. *Journal of Computer and System Sciences*, 15(2):213–231, 1977. doi:10.1016/S0022-0000(77)80007-X.

31 T. Toffoli and N. Margolus. *Cellular Automata Machines*. MIT Press, 1987.

32 T. Toffoli and N. Margolus. Invertible cellular automata: A review. *Physica D: Nonlinear Phenomena*, 45(1–3):229–253, 1990. doi:10.1016/0167-2789(90)90185-R.

33 A. Toom. Stable and attractive trajectories in multicomponent systems. In R. L. Dobrushin and Ya. G. Sinai, editors, *Multicomponent Random Systems*, pages 549–575. Marcel Dekker, 1980.

34 A. L. Toom, N. B. Vasilyev, O. N. Stavskaya, L. G. Mityushin, G. L. Kuryumov, and S. A. Pirogov. Discrete local Markov systems. In R. L. Dobrushin, V. I. Kryukov, and A. L. Toom, editors, *Stochastic cellular systems: ergodicity, memory, morphogenesis*. Manchester University Press, 1990.

**35** G. Y. Vichniac. Simulating physics with cellular automata. *Physica D: Nonlinear Phenomena*, 10(1–2):96–116, 1984. doi:10.1016/0167-2789(84)90253-7.

**36** J. von Neumann. Probabilistic logics and the synthesis of reliable organisms from unreliable components. In C. E. Shannon and J. McCarthy, editors, *Automata Studies*, pages 43–98. Princeton University Press, 1956. doi:10.1515/9781400882618.

# Fixed Point Constructions in Tilings and Cellular Automata

**Ilkka Törmä**
Department of Mathematics and Statistics, University of Turku, Finland

## Abstract

The fixed point construction is a method for designing tile sets and cellular automata with highly nontrivial dynamical and computational properties. It produces an infinite hierarchy of systems where each layer simulates the next one. The simulations are implemented entirely by computations of Turing machines embedded in the tilings or spacetime diagrams. We present an overview of the construction and list its applications in the literature.

**2012 ACM Subject Classification** Theory of computation → Models of computation

**Keywords and phrases** Tilings, Wang tiles, cellular automata, multidimensional symbolic dynamics, self-simulation

**Digital Object Identifier** 10.4230/OASIcs.AUTOMATA.2021.4

**Category** Invited Talk

**Related Version** *Full Version*: `https://arxiv.org/abs/2105.00443`

## 1 Introduction

In this short survey we present an overview of the *fixed point construction* of Wang tile sets and cellular automata, also known as *programmatic self-simulation*. It is a method of defining an infinite sequence of tile sets $(T_k)_{k\geq 0}$ with the property that each $T_k$ simulates $T_{k+1}$, in the sense that valid tilings over $T_k$ have the form of infinite regular $N_k \times N_k$ grids for some constant $N_k > 1$ and each grid cell behaves like a tile of $T_{k+1}$. The simplest variant has $T_k = T_{k+1}$ for all $k$, so that $T_0$ simulates itself. The defining property of the technique is that the simulations are implemented almost entirely "in software" by computations of embedded universal Turing machines. This provides considerable flexibility, since modifying the next tile set $T_{k+1}$ is as easy as modifying the program of the machine that runs on $T_k$. On the flip side, the flexibility comes with the price of nonmodularity, as nontrivial constructions are hard to reuse without presenting them in detail. The tile sets produced by the fixed point method are also inevitably massive, and it is usually impractical to determine them exactly. Of course, this is true for most constructions in symbolic dynamics that involve simulating nontrivial computation, but the fixed point construction stands out by requiring it even for results that do not have a computational flavor, such as the existence of an aperiodic tile set.

The fixed point construction has its roots in Kurdyumov's informal article [24] on probabilistic cellular automata. Kurdyumov's ideas were implemented rigorously by Gács in his disproof of the positive rates conjecture [13, 14], which uses programmatic self-simulation as the basis of an extremely intricate construction. Durand, Romashchenko and Shen isolated this part of the construction and applied it to Wang tile sets in a series of papers [8, 9, 10, 11] culminating in [12]. A series of further applications by them and other authors has followed. Interestingly, many of these share a common theme: a result on tiling systems, sofic shifts or cellular automata was proved in all dimensions except one or two using a geometric construction, such as Robinson tiles [28] or Mozes's realization of substitutive shifts [27], and the remaining low-dimensional cases were settled by a fixed point construction.

27th IFIP WG 1.5 International Workshop on Cellular Automata and Discrete Complex Systems (AUTOMATA 2021).
Editors: Alonso Castillo-Ramirez, Pierre Guillon, and Kévin Perrot; Article No. 4; pp. 4:1–4:13

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

## 2 Definitions and notation

Let $A$ be a finite alphabet and $d \geq 1$. A *pattern* is a function $P : D \to A$ with *domain* $D = D(P) \subseteq \mathbb{Z}^d$. Patterns of domain $\mathbb{Z}^d$ are called *configurations*, and they form the *$d$-dimensional full shift* $A^{\mathbb{Z}^d}$. Define the *shift map* $\tau : \mathbb{Z}^d \times A^{\mathbb{Z}^d} \to A^{\mathbb{Z}^d}$ by $(\tau^{\vec{n}}x)_{\vec{v}} = x_{\vec{v}+\vec{n}}$. A set of finite patterns $F$ defines a *subshift* as the subset $X_F = \{x \in A^{\mathbb{Z}^d} \mid \forall P \in F, \vec{v} \in \mathbb{Z}^d : (\tau^{\vec{v}}x)|_{D(P)} \neq P\}$ where no pattern from $F$ occurs at any position. If $F$ is finite, $X_F$ is a *shift of finite type* (SFT), and if $F$ is computably enumerable, $X_F$ is an *effective subshift*. A pattern is *locally valid* for $F$ if no element of $F$ occurs in it as a sub-pattern. The set of domain-$D$ patterns occurring in a subshift $X$ is denoted $\mathcal{L}_X(D) = \{x|_D \mid x \in X\}$. If a subshift $X$ does not properly contain another nonempty subshift, it is *minimal*. If a subshift $X$ is a union of minimal subshifts, it is *quasiperiodic*.[1]

A particular class of 2-dimensional SFTs is given by sets of *Wang tiles*, which are squares with colored edges. Formally, a Wang tile set is a subset of $C^4$ for a finite set $C$ of colors, with the four components representing the colors of the east, north, west and south edges of the square. We forbid exactly those $1 \times 2$ and $2 \times 1$ patterns where the adjacent edges of the two tiles have different colors. A $d$-dimensional SFT $X$ is *aperiodic* if for all $x \in X$ and $\vec{p} \in \mathbb{Z}^d$ there exists $\vec{v} \in \mathbb{Z}^d$ with $x_{\vec{v}+\vec{p}} \neq x_{\vec{v}}$. A set of Wang tiles is called aperiodic if it defines a nonempty aperiodic SFT.

A function $\phi : X \to Y$ between subshifts $X \subseteq A^{\mathbb{Z}^d}, Y \subseteq B^{\mathbb{Z}^d}$ is a *block map* if there is a finite *neighborhood* $N \subset \mathbb{Z}^d$ and *local function* $\Phi : \mathcal{L}_X(N) \to B$ such that $\phi(x)_{\vec{v}} = \Phi((\tau^{\vec{v}}x)|_N)$ for all $x \in X$ and $\vec{v} \in \mathbb{Z}^d$. If $X$ is an SFT and $\phi$ is surjective, then $Y$ is a *sofic shift* and $X$ is an *SFT cover* of $Y$. A *cellular automaton* (CA) is a block map from a full shift to itself. A *spacetime diagram* of a CA $\phi : A^{\mathbb{Z}^d} \to A^{\mathbb{Z}^d}$ is a configuration $x \in A^{\mathbb{Z}^{d+1}}$ with $x|_{\mathbb{Z}^d \times \{i+1\}} = \phi(x|_{\mathbb{Z}^d \times \{i\}})$ for each $i \in \mathbb{Z}$. In a spacetime diagram, the $d$-dimensional slices obtained by fixing the last coordinate form a trajectory of $\phi$.

A standard reference for one-dimensional subshifts, with a short appendix on the multidimensional setting, is [25]. See [23] for a survey on the theory of CA. Wang tiles were first defined in [32], and aperiodic sets of Wang tiles were first constructed in [2].

## 3 The fixed point construction

### 3.1 Tile sets

In this section we present an outline of the fixed point construction in the context of Wang tiles following [12]. We omit much of the detail.

▶ **Definition 1.** *Let $T$ and $S$ be two sets of Wang tiles. A* simulation *of $S$ by $T$ with* zoom factor *$N$ is an injective function $\alpha : S \to T^{N \times N}$ such that:*

- *For any $s_1, s_2 \in S$, the horizontal concatenation $s_1 s_2$ is locally valid over $T$ if and only if $\alpha(s_1)\alpha(s_2)$ is locally valid over $S$, and similarly for their vertical concatenation.*
- *For every valid tiling $x \in T^{\mathbb{Z}^2}$, there is a unique vector $\vec{v} \in [0, N-1]^2$ such that $(\tau^{\vec{v}+(iN,jN)}x)|_{[0,N-1]^2} \in \alpha(S)$ for all $(i, j) \in \mathbb{Z}^2$.*

Suppose first that we are given the tile set $S$ and wish to define, for each large enough zoom factor $N$, a tile set $T = T(N)$ and a simulation $\alpha : S \to T^{N \times N}$. We will implement $T$ in a way that makes the choice $S = T$ possible with relatively minor modifications. Assume

[1] Note that an arbitrary union of minimal subshifts is generally not a subshift.

the colors of $S$-tiles are binary strings of some length $k$, that is, elements of $\{0,1\}^k$. First, each tile $T$ has an *address* $(i,j) \in [0,N-1]^2$. Local rules ensure that its east and north neighbors have addresses $(i+1,j)$ and $(i,j+1)$ modulo $N$, respectively. Concretely, the set of colors of $T$ is $[0,N-1] \times C$ for some auxiliary set $C$, and each tile with address $(i,j)$ has the form $((j+1,c_e),(i+1,c_n),(j,c_w),(i,c_s))$ for some $c_e,c_n,c_w,c_s \in C$. Then in a valid tiling $x \in T^{\mathbb{Z}^2}$, the tiles are partitioned into a grid of $N \times N$ blocks called *macrotiles* whose southwest corners are the tiles with address $(0,0)$. The address of each tile is its relative position within the macrotile containing it.

Each tile of a macrotile $t \in T^{N \times N}$ has a specific role that depends on its address. A fixed rectangular subset $R = [a,b] \times [c,d] \subset [0,N-1]^2$ of addresses forms the *computation zone*. It stores a simulated computation of a Turing machine $M$, which recognizes the set $S$ in the sense that it halts on input $w \in \{0,1\}^{4k}$ if and only if $w$ encodes a tile of $S$. A reasonable choice for the parameters is $a = c = \lfloor N/3 \rfloor$ and $b = d = \lfloor 2N/3 \rfloor$, so that the dimensions of $R$ increase linearly in $N$. The simulation of $M$ is implemented in some standard way, such as the one presented in [28]. The bottom row of $R$ initializes the computation, and each row above it simulates one computation step. If $M$ does not halt before reaching the top row of $R$, a tiling error is produced. As long as $N$ (and thus $R$) is large enough, the simulated machine has enough time and space to verify any input that encodes a tile of $S$.

Fix an interval $I \subset [0,N-1]$ of length $k$, the number of bits in a color of $S$. Each tile with address in $B = (\{0,N-1\} \times I) \cup (I \times \{0,N-1\})$ stores an additional bit, called a *border bit*. Their purpose is to encode the four colors of the simulated tile of $S$. The border bits are constrained to be equal to those of the neighboring macrotiles. For each address in $B$, we fix a contiguous path of addresses to the bottom row of the computation cone, making sure that paths originating from distinct addresses are pairwise disjoint. Using local rules, we force all tiles with addresses on a given path to store the same bit. In this way the border bits are routed to the computation zone, and the resulting binary word of length $4k$ forms the input to the simulated machine $M$. If $M$ accepts this word, then it represents a tile $s \in S$, and we set $\alpha(s) = t$. With this construction the set of locally valid macrotiles equals $\alpha(S)$, and the adjacency rules of macrotiles are identical to those of $S$. Thus $T$ simulates $S$ with zoom factor $N$.

We would now like to choose $S = T$, so that $T = T(N)$ simulates itself. There are a few complications that we need to handle. First, the colors of $T$ must be binary words of constant length. We encode one component of the address in $\lceil \log_2 N \rceil$ bits and the remaining data in another $m = \lceil \log_2 |C| \rceil$ bits. Second, the state set and transition rules of the Turing machine $M$, which are part of the tile set $T$, depend on $S$. We replace $M$ by a fixed universal Turing machine $M_{\mathrm{U}}$ that takes as input a binary word $w \in \{0,1\}^*$ and, on a separate track of the tape, a program $p \in \{0,1\}^*$, and computes $p(w)$. The machine $M_{\mathrm{U}}$ should also be efficient in the sense that for any Turing machine $M'$, there is a program $p$ such that $M_{\mathrm{U}}(p,w) = M'(w)$ for all $w$ and the number of computation steps in $M_{\mathrm{U}}(p,w)$ is polynomial in $|p|$, $|w|$, and the number of computation steps in $M'(w)$. Now the set $C$ of auxiliary colors of $T(N)$ (and thus the number $m$) is fixed, and to simulate $T(N)$ itself, it suffices to find a suitable program $p_T = p_{T(N)}$. We enforce the simulation by requiring that each tile with address $(a+i,b)$ stores the bit $(p_T)_i$ on the program track of the simulated tape of $M_{\mathrm{U}}$, where $(a,b)$ is the southeast corner of the computation zone. We call this *the program condition*, and it is part of the definition of $T$.

Since we introduced the program condition into $T$, we must enforce it in the simulated version as well. Namely, suppose for a moment that the machine $M_{\mathrm{U}}$ always has enough time and space to finish its simulated computation in a macrotile. At this point of the construction

we then have a tile set $T$ that simulates "tile set $T$ without the program condition," which is of course not equal to $T$. We can enforce the program condition in the simulated tiling as well by modifying the program $p_T$ as follows. If the machine $M_{\mathrm{U}}$ simulated inside a macrotile $t \in T^{N \times N}$ is given inputs that correspond to a tile with address $(a+i, b)$ on the bottom row of the computation zone, then it reads the program bit $d \in \{0,1\}$ of any tile of $t$ with address $(a+i, b')$ for $b' \in [0, N-1]$ (which are all equal), and verifies that the simulated program bit of $t$ equals $d$. The program condition of $T$ ensures that the initial tape of the machine $M_{\mathrm{U}}$ contains the program $p_T$, so we are guaranteed that $d = (p_T)_i$. After this modification, $T$ simulates itself (given the assumption on $M_{\mathrm{U}}$).

Finally, consider the time and space requirements of $M_{\mathrm{U}}$. The number of bits needed to store a color of $T(N)$ is $\ell = \lceil \log_2 N \rceil + m = O(\log N)$, since $m$ is fixed. The program $p_T$ can store the number $N$ in a variable, so its length is likewise $O(\log N)$. It should be clear that for each $w \in \{0,1\}^{4\ell}$ the computation of $p_T(w)$ runs in time and space $O(\mathrm{poly}(\log N))$ as long as we implement $T$ and $p_T$ in a reasonable way. For large enough $N$ the computation zone can accommodate the computation, and then $T = T(N)$ is a self-simulating tile set. Each valid tiling over $T$ is divided into macrotiles of level 1, which form macrotiles of level 2, which form macrotiles of level 3, and so on. In particular, the tile set is aperiodic. In the standard terminology, a level-$k$ macrotile is a *child* of the level-$(k+1)$ macrotile that contains it.

The simulation function $\alpha : T \to T^{N \times N}$ can be seen as a two-dimensional *substitution*. We can iterate it to obtain substitutions $\alpha^k : T \to T^{N^k \times N^k}$ for $k \geq 0$, and define the *substitutive subshift* $X_\alpha \subset T^{\mathbb{Z}^2}$ by forbidding all finite patterns that do not occur in any $\alpha^k(t)$ for $k \geq 0$ and $t \in T$. The set of valid tilings over $T$ contains $X_\alpha$, and $T$ can quite easily be modified to guarantee that these subshifts are equal. In [7], Durand and Romashchenko presented a variant of the construction in which $\alpha$ is *primitive*, meaning that for some $k \geq 0$, every $t \in T$ occurs in $\alpha^k(s)$ for every $s \in T$. In this case $X_\alpha$ is a minimal subshift. We will discuss their results more thoroughly in Section 4.2.

## 3.2 Cellular automata

The main idea of the fixed point construction for one-dimensional cellular automata is the same as for tile sets, and in fact the "good" spacetime diagrams of a fixed point CA have a similar substitutive structure to those of fixed point tilings, except that the width and height of the macrotiles are distinct. The usual terminology is somewhat different, as is the implementation of information transfer between macrotiles. As before, we first construct a CA $\phi : A^{\mathbb{Z}} \to A^{\mathbb{Z}}$ that simulates a given CA $\psi : B^{\mathbb{Z}} \to B^{\mathbb{Z}}$. The notion of simulation has long been used informally in CA literature, but for the sake of being exact, let us use a version of *injective bulking* as studied in [5].

▶ **Definition 2.** *A cellular automaton $\phi : A^{\mathbb{Z}} \to A^{\mathbb{Z}}$* simulates *another CA $\psi : B^{\mathbb{Z}} \to B^{\mathbb{Z}}$ if there exist integers $Q, U \geq 1$ and $s \in [0, Q-1]$ and an injection $\alpha : B \to A^Q$ such that $\phi^U(\tau^s(\alpha(x))) = \alpha(\psi(x))$ for all $x \in B^{\mathbb{Z}}$. Here $\alpha(x) \in A^{\mathbb{Z}}$ is defined by concatenating the blocks $\alpha(x_i)$ for $i \in \mathbb{Z}$.*

We assume that $\psi$ has neighborhood $N = \{-1, 0, 1\}$ and its state set consists of binary words of constant length $k$, and construct $\phi$ so that it also has neighborhood $N$. The simulation parameters $Q < U$ are called the *colony size* and *work period*, and we choose $s = 0$. The work period $U$ is typically vastly larger than $Q$, but it is possible to implement the CA so that $Q/U$ is arbitrarily close to 1. Every state of $\phi$ consists of an element of $[0, Q-1] \times [0, U-1]$

called its *address* and *age*[2] plus some additional data. Say that a configuration $x \in A^{\mathbb{Z}}$ is *valid at* $i \in \mathbb{Z}$ if the states $x_{i-1}x_ix_{i+1}$ have addresses $j, j+1, j+2 \bmod Q$ for some $j$ and their ages are equal. We only define the prototypical CA $\phi$ on configurations that are valid everywhere. In this case, the address of a cell never changes and its age increases by 1 modulo $U$ on every time step.

A *colony* is a sequence of $Q$ adjacent cells with addresses $0, 1, 2, \ldots, Q-1$ and equal ages. Each cell of a colony stores a *simulation bit*, and the $k$ leftmost bits form the simulated state of the colony. The "life cycle" of a colony, usually called a *work period* when it cannot be confused with $U$, begins at age 0 and ends at age $U-1$. At the first step of the work period when each cell has age 0, the simulation bits of the cells are copied onto two separate tracks called the *left and right mailboxes*. If the age of a cell is between 1 and $Q$, it copies the contents of these tracks from its left and right neighbors. Otherwise they remain unchanged. In this way, during the first $Q+1$ steps of its life cycle every colony transmits its simulation bits to its nearest neighbors. The details of this process vary between implementations. Here we have followed [31] and in particular assumed that $\phi$ is synchronous and deterministic. If it is asynchronous or subject to local errors, a different scheme is needed to minimize loss of transmitted data.

Each cell of a colony also stores a *program bit*, and these bits together form a program $p \in \{0,1\}^*$. At age $Q+1$, the leftmost cell of each colony initializes the computation of the efficient universal Turing machine $M_{\mathrm{U}}$. The simulated head of $M_{\mathrm{U}}$ is usually called the *agent*. The agent scans the program bits, simulation bits and mailboxes of the colony, computes a new simulated state $w \in \{0,1\}^k$ according to the program $p$, and writes $w$ onto the simulation bits of the $k$ leftmost cells. We assume that the agent never steps outside the colony and halts before the work period ends at age $U-1$. Then the age of the colony resets to 0 and its life cycle begins anew.

As in the previous section, the CA $\phi$ simulates $\psi$ as long as $Q$ and $U$ are large enough to satisfy the time and space requirements of $M_{\mathrm{U}}$. A suitable choice of the parameters allows self-simulation, meaning that we can choose $\phi = \psi$. Note that a correct simulation hierarchy requires a program $p_\phi$ implementing $\phi$ to be stored in every colony of the initial configuration, on every simulation level. On the first level this can be enforced by restricting the state set of $\phi$ analogously to the program condition of the tile set construction, and then $M_{\mathrm{U}}$ can check whether the next simulation level uses an identical program. It depends entirely on the application how the CA behaves if the programs do not match, or if the simulation structure is invalid in some other way: it can try to correct the error, produce a special error state, or have only a partially defined local rule.

## 3.3 Variable zoom factor and communication between simulation levels

The construction is flexible enough that instead of having the tile set or the CA simulate itself, we can modify the simulated system depending on the simulation level, resulting in a sequence of tile sets $(T_k)_{k\geq 0}$ or automata $(\phi_k)_{k\geq 0}$. The motivation is that the tile set or CA has some desirable property $P(N)$ that is limited by the zoom factor $N$, and by increasing the zoom factor on consecutive simulation levels it may be possible to obtain $P(N)$ for all $N \in \mathbb{N}$. A common use case is that $P(N)$ has a computational flavor, such as tiling the plane only if a fixed Turing machine halts within $N$ steps.

[2] Notice the difference in terminology to the tile set construction, where *address* referred to a two-dimensional vector.

We discuss the implementation only in the case of tile sets, as the same ideas apply to the CA construction. Our goal is then to define a sequence $(T_k, N_k)_{k\geq 0}$ of tile sets and numbers such that each tile set $T_k$ simulates $T_{k+1}$ with zoom factor $N_k$, and we would like to have as much freedom as possible in choosing the latter. For this, we modify the construction of Section 3.1 as follows. The program $p_T$ has an additional integer parameter: $p_T(k, w)$ determines if $w$ encodes a tile of $T_k$. The program condition is extended so that in addition to the program $p_T$, each tile of $T_0$ stores the number 0. For each $k \geq 0$, the tile set $T_k$ implements a variant of the program condition where each tile of $T_{k+1}$ stores $p_T$ and the number $k + 1$. This ensures that all macrotiles have access to their simulation level.

The constraints on the zoom factors are slightly different than in the uniform case. First, both $N_{k+1}$ and $T_{k+1}$ should be computable from $k$ in time and space $O(N_k)$. Second, $N_k = \Omega(\log N_{k+1} + \log k)$ since the binary representation of any address in $[0, N_{k+1} - 1]^2$ and the simulation level $k$ should fit in any level-$k$ macrotile. To get a sense of the class of allowed sequences, $N_k = \lceil \log k \rceil$ for large enough $k$ grows too slowly, while $N_{k+1} = 2^{N_k}$ grows too fast. The choices $N_k = \Theta(k^a)$ for $a \in \mathbb{Q}_{>0}$, $N_k = \Theta(2^k)$ and $N_k = \Theta(2^{2^{2^k}})$ are all acceptable.

In most applications, each macrotile stores some amount of additional information not listed above, and there are additional relations between the data stored by macrotiles of consecutive levels. In some sense, a macrotile can "communicate" with its children. The technical details vary, but the main idea is that each level-$k$ macrotile contains a special field in its state set which the simulated machine $M_{\mathrm{U}}$ of its parent is allowed to access. The parent may thus read an arbitrary "message" from a subset of macrotiles on the bottom row of the computation zone. The children can synchronize this message among themselves using local rules, so that it is visible to all of them or some convenient subset. In the CA context, the agent of a colony can read and modify any data that was stored in the children during the last work period, so inter-level communication is slightly easier to implement.

# 4 History and applications

## 4.1 Fault-tolerant cellular automata

The fixed point construction was first introduced in the context of *probabilistic cellular automata* (PCA). We give some basic definitions; see [26] for a more thorough review on the subject. A $d$-dimensional PCA $\phi$ on $A^{\mathbb{Z}^d}$ consists of a neighborhood $N \subset \mathbb{Z}^d$ and a local rule $\Phi : A^N \to \mathbb{P}(A)$, where $\mathbb{P}(A)$ is the set of probability distributions over $A$. The local rule is applied to every coordinate of a configuration $x \in A^{\mathbb{Z}^d}$ as in the case of deterministic CA, but the result is the probability distribution $\phi(x) \in \mathbb{P}(A^{\mathbb{Z}^d})$ where the value of each cell $\phi(x)_{\vec{v}}$ is drawn independently from $\Phi((\tau^{\vec{v}}x)|_N)$. In this way $\phi$ lifts to a function $\phi : \mathbb{P}(A^{\mathbb{Z}^d}) \to \mathbb{P}(A^{\mathbb{Z}^d})$ on distributions, denoted $\mu \mapsto \mu\phi$. Every PCA has at least one fixed point, called an *invariant measure*. If $\phi$ has a unique invariant measure $\mu$ and for every other measure $\nu$ the sequence $(\nu\phi^n)_{n\in\mathbb{N}}$ converges weakly to $\mu$, then $\phi$ is *ergodic*. Intuitively, an ergodic PCA gradually forgets all details about its initial state. We say $\phi$ has *positive rates* if the probability of $\Phi(P) = a$ is positive for all $P \in A^N$ and $a \in A$. In other words, every local transformation occurs under $\phi$ with positive (but possibly very small) probability. The *positive rates conjecture* states that if a one-dimensional PCA has positive rates, then it must be ergodic.

In 1986 (and with an expanded paper in 2001), Gács disproved the conjecture by an intricate construction of a PCA that behaves like a deterministic CA except that at each time step, each cell has an extremely small but positive probability of "making an error" and ending up in an arbitrary state [13, 14]. The design is based on a short paper of Kurdyumov from 1978 that likewise claims to refute the conjecture, but lacks detail [24].

▶ **Theorem 3** (Gács). *There exists a nonergodic 1D PCA with positive rates.*

The construction uses self-simulation to recognize and correct clusters of errors that occur at various scales. Non-ergodicity is achieved by defining the state set as a product $\{0,1\} \times A$, initializing each cell in some state $x_i \in \{b\} \times A$ for the same bit $b$, and "remembering" the bit in the sense that the probability of seeing the opposite bit $1-b$ in any given cell at any given time is low.[3] If the error rate is small enough, an argument from percolation theory shows that errors will almost surely occur in semi-isolated "bursts" that fit in finite (but unbounded) space-time rectangles. At the lowest simulation level, each cell contains information about its own simulation values (address, age...) and the bit to remember, as well as those of its nearest neighbors, and isolated individual errors can be fixed in one time step by a majority vote between three cells. Larger bursts of errors are detected and repaired by higher-level colonies. One of the main difficulties is that long sequences of errors can produce a segment of states that resembles a valid simulation structure. Its incompatibility with the global hierarchy is only visible on its endpoints, where it cannot be determined locally which side belongs to the erroneous segment. See Gray's condensed version of the construction for more (but not nearly all) technical details [16]. Gács and Çapuni have applied the same ideas to fault-tolerant Turing machines [4].

## 4.2 Realizations by sofic shifts

In [20], Hochman proved that $d$-dimensional effectively closed subshifts can be realized as the *projective sybdynamics* of $(d+2)$-dimensional sofic shifts, that is, sets of $d$-dimensional hyperplanes occurring in them. The construction uses the older result of Mozes that two-dimensional substitutive subshifts are sofic [27] to divide each configuration into regions of increasing size containing simulations of Turing machines. The dimension was reduced from $d+2$ to $d+1$ independently by Aubrun and Sablik in [1] with Robinson tiles [28], and by Durand, Romashchenko and Shen in [12] using fixed point tilings. It has since been used as a "black box" in numerous other constructions. Incidentally, the realization result of Mozes was also reproved with a fixed point construction in [12].

▶ **Theorem 4** (Aubrun & Sablik; Durand & Romashchenko & Shen). *Let $X \subseteq A^{\mathbb{Z}^d}$ be an effectively closed subshift. Then the higher-dimensional subshift $Y = \{y \in A^{\mathbb{Z}^{d+1}} \mid \exists x \in X : \forall \vec{w} \in \mathbb{Z}^{d+1} : y_{\vec{w}} = x_{\pi_d(\vec{w})}\}$ is sofic, where $\pi_d : \mathbb{Z}^{d+1} \to \mathbb{Z}^d$ is defined by $\pi_d(i_1, \dots, i_{d+1}) = (i_1, \dots, i_d)$.*

The article [12] is a culmination of a series of shorter papers by the same authors. It presents the prototypical fixed point construction, collects their previous results into one place, and extends some of them. In their proof of Theorem 4 (in the case $d = 1$), there is a separate layer of $A$-tiles in which each vertical stripe is tiled with copies of a single $A$-symbol. The simulations have zoom factors $N_k = Q^{c^k}$ for some large constant $Q$ and $c > 2$. A macrotile of level $k$ with address $(x, y)$, where $y < \prod_{i=0}^{k} N_i$, is tasked with retrieving and storing the symbols of $f(k)$ consecutive columns of the $A$-layer situated $y$ steps away from its left border, where $f : \mathbb{N} \to \mathbb{N}$ is a very slowly growing function. The definition of $N_k$ ensures that $N_{k+1} \ll \prod_{i=0}^{k} N_i$, so that in a level-$(k+1)$ macrotile $t$, every length-$f(k)$ word of the $A$-layer occurring within it (or right at the border between it and its neighbor) is stored in at least one child. Using a series of inter-level messages, $t$ is in turn able to retrieve

[3] In fact, the construction can remember an infinite sequence of bits.

its designated length-$f(k+1)$ word. It also computes a prefix of the sequence of forbidden patterns that defines $X$ and verifies that none of them occur in the word it retrieved. As a direct application, the authors reprove the main result of [6]: there exists a tile set where the Kolmogorov complexity of every $N \times N$ pattern that occurs in a tiling is $\Omega(N)$, which is asymptotically optimal for an SFT. Namely, we can define $X \subset \{0,1\}^{\mathbb{Z}}$ by forbidding all words $w$ whose Kolmogorov complexity is below $a|w| - b$ for suitable constants $a, b > 0$ and apply Theorem 4.

Some dynamical properties of the subshift $X$ in Theorem 4 can be lifted to the SFT cover $Z$ of $Y$. One of the main results of Durand and Romashchenko in [7] states that if $X$ is minimal or quasiperiodic, then the same property can be imposed on $Z$. They modify the construction so that every macrotile contains a zone of "diversification slots" where all valid $2 \times 2$ patterns of tiles are manually forced to appear. In the minimal case, there is the additional complication that if the positions of the macrotile grids and the layer of $A$-tiles in the SFT cover can be independent, the result is generally not minimal. This is overcome by choosing a canonical configuration from each and implementing the minimal shift generated by their combination. In the same article, the authors characterize the Turing degree spectra of quasiperiodic SFTs, which are exactly the upper closures of effectively closed subsets of $\{0,1\}^{\mathbb{N}}$ (see [7, Theorem 4] for details). In particular, there exist nonempty quasiperiodic SFTs with only noncomputable configurations.

In [33], Westrick proves two sofic realization results that utilize the fixed point construction. They concern two-dimensional binary subshifts where every connected component of 1-cells is a (possibly degenerate) square.

▶ **Theorem 5** (Westrick). *Let $X_{\mathrm{sq}} \subseteq \{0,1\}^{\mathbb{Z}^2}$ be the 2D subshift where every connected component of 1-cells is either a finite square, a quarter-plane, a half-plane, or the entire $\mathbb{Z}^2$.*

**(a)** *Every effectively closed subshift $X \subseteq X_{\mathrm{sq}}$ that forbids all pairs of distinct squares of identical size (and possibly other patterns) is sofic.*

**(b)** *For $S \subseteq \mathbb{N}$, let $X_S \subseteq X_{\mathrm{sq}}$ be the subshift that forbids the $n \times n$ squares for all $n \in \mathbb{N} \setminus S$. If $S$ is computably enumerable, then $X_S$ is sofic.*

We only sketch the construction proving the first result, as the other is quite similar. The subshift $X_{\mathrm{sq}}$ is easily seen to be sofic, so we focus on the additional constraints of $X$. Westrick uses the variable zoom factors $N_k = 2^{2^{2^k}}$. In addition to the simulation structure, each macrotile $t$ stores the following information: first, the size and relative position of each finite square of 1-cells that has at least one side completely inside $t$ or one of its neighbors; second, the orientation and relative position of each corner and horizontal or vertical boundary of 1-cells in $t$ or its neighbors that if not part of an already stored finite square; third, auxiliary bits that help enforce the consistency of this information between $t$ and its parent.

If the sizes of all squares are distinct, then each level-$k$ macrotile can contain at most $O(L_k^{2/3})$ squares, where $L_k = \prod_{i=0}^{k} N_k$ is its side length, and their sizes and positions can be encoded in $O(L_k^{2/3} \log L_k)$ bits. This data is small enough to fit in the macrotile, but transporting it from its children to the computation zone is nontrivial, since it is too large to be simply broadcast to each child. Westrick uses a graph theoretical argument to show that one can route the information in a nondeterministic way. Additionally, the universal Turing machine $M_{\mathrm{U}}$ needs to be replaced by a multi-tape machine to speed up the computation of the relative offsets of the squares. As in Theorem 4, the number of additional time steps used for computing the forbidden patterns of $X$ is increased at each simulation level. The patterns formed by small squares are checked first, so any forbidden pattern will be eventually discovered and leads to a tiling error.

## 4.3 Robust tilings

In addition to Theorem 4 and related realization results, the paper [12] presents tile sets that are *robust with respect to random errors*, meaning that a configuration that is locally valid on a large subset of $\mathbb{Z}^2$ is guaranteed to be equal to a single valid tiling in a large[4] subset of $\mathbb{Z}^2$. In fact, the main result is the construction of a tile set that combines robustness with the aforementioned property that every tiling is algorithmically complex.

▶ **Definition 6.** *Let $T$ be a set of Wang tiles and $E \subseteq \mathbb{Z}^2$. A $(T, E)$-tiling is a configuration $x \in T^{\mathbb{Z}^2}$ such that for all neighboring coordinates $\vec{v}, \vec{w} \in \mathbb{Z}^2 \setminus E$, the colors at the common border of $x_{\vec{v}}$ and $x_{\vec{w}}$ match. The Bernoulli distribution on subsets of $\mathbb{Z}^2$ with $\mathbb{P}(\vec{v} \in E) = \epsilon$ for each $\vec{v}$ is denoted $B_\epsilon$.*

▶ **Theorem 7** (Durand & Romashchenko & Shen)**.** *There exists a tile set $T$ that tiles the plane and $a, b > 0$ with the following properties. In every valid $x \in T^{\mathbb{Z}^2}$, every $n \times n$ square has Kolmogorov complexity at least $an - b$. For small enough $\epsilon > 0$, for $B_\epsilon$-almost every $E \subseteq \mathbb{Z}^2$, for each $(T, E)$-tiling $x$, the Kolmogorov complexity of $x_{[-n,n]^2}$ is $\Omega(n)$, and there is a valid tiling $y \in T^{\mathbb{Z}^2}$ with $\limsup_n |\{\vec{v} \in [-n, n]^2 \mid x_{\vec{v}} \neq y_{\vec{v}}\}|/(2n + 1)^2 \leq 1/10$.*

The construction of robust self-simulating tile sets turns out to be much simpler than the error-correcting CA discussed in Section 4.1. Similarly to that construction, if $\epsilon$ is small enough, then almost all error sets $E$ can be decomposed into isolated finite subsets that can be handled separately. It is then enough to modify the prototypical fixed point construction so that every locally valid pattern shaped like the $5 \times 5$ annulus $[-2, 2]^2 \setminus [-1, 1]^2$ can be extended into a valid $5 \times 5$ rectangle in a unique way. Namely, this property is inherited by macrotiles of all simulation levels, which can then correct arbitrarily large (but finite) incorrect patterns. The additional condition on the Kolmogorov complexity of $(T, E)$-tilings requires a little more machinery, since the uniformity of the vertical columns that store the bits of the high-complexity shift $X$ can be broken by the error set $E$. This is remedied by duplicating the information in several columns of macrotiles and periodically checking it against an error-correcting code.

## 4.4 Unique ergodicity

The article [31] by Törmä is an application of the fixed point construction to the dynamics of cellular automata. It concerns a weakening of the notion of *nilpotency*.

▶ **Definition 8.** *A CA $\phi : A^{\mathbb{Z}^d} \to A^{\mathbb{Z}^d}$ with a quiescent state $0 \in A$ is* nilpotent *if there exists $n \in \mathbb{N}$ such that $\phi^n(A^{\mathbb{Z}^d}) = \{0^{\mathbb{Z}^d}\}$. It is* asymptotically nilpotent *if for each $x \in A^{\mathbb{Z}^d}$ we have $\phi^n(x) \to 0^{\mathbb{Z}^d}$ in the product topology as $n \to \infty$. It is* uniquely ergodic *if it has a unique invariant measure.*

A nilpotent CA sends every initial configuration into the same uniform "sink" in a bounded number of steps. Asymptotic nilpotency means that for every initial configuration, each cell will be in a nonzero state on only a finite (but not necessarily uniform) number of time steps. It was shown in [17] that asymptotically nilpotent one-dimensional CA are in fact nilpotent, and the result was generalized in [29, 30]. As for unique ergodicity, note that since 0 is quiescent, the unique invariant measure cannot be any other than the Dirac

[4] Large in a slightly different sense: the first subset comes from a measure-1 set with respect to a biased Bernoulli distribution, the second has asymptotic density close to 1.

point measure concentrated on $0^{\mathbb{Z}^d}$. Then by classical results of ergodic theory, $\phi$ is uniquely ergodic if and only if for every initial configuration, the asymptotic proportion of time steps that any given cell spends in a nonzero state equals zero. The main result of [31] states that even one-dimensional uniquely ergodic CA are not necessarily nilpotent.

▶ **Theorem 9** (Törmä). *There exists a uniquely ergodic CA $\phi : A^{\mathbb{Z}} \to A^{\mathbb{Z}}$ that has a quiescent state and is not nilpotent.*

This CA is defined by a fixed point construction with variable zoom factors. The alphabet of the CA $\phi_k$ on each simulation level $k$ contains a designated "blank" state $B_k$, with $B_0 = 0$. Each cell of a level-$k$ colony will spend a proportion $p_k$ of its work period simulating the state $B_k$, with $p_k \to 1$ as $k \to \infty$. If a colony and both of its neighbors simulate the state $B_k$, it is temporarily replaced by a segment of 0-cells. Any local inconsistency in the simulation structure causes all nearby colonies to assume the blank state. These properties ensure that every locally correct simulation structure is asymptotically infinitely sparse, and locally incorrect structures are quickly destroyed. On the other hand, since a carefully chosen initial condition gives rise to an infinite nested simulation, the CA is not nilpotent.

## 4.5 Realization of topological entropy

The *topological entropy* of a topological dynamical system measures its complexity and unpredictability. In the case of subshifts and cellular automata, it can be characterized as a limit of counting patterns of increasing sizes that occur in the valid configurations and spacetime diagrams.

▶ **Definition 10.** *The topological entropy of a subshift $X \subseteq A^{\mathbb{Z}^d}$ is the limit $h(X) = \lim_{n\to\infty} n^{-d} \log |\{x_{[0,n-1]^d} \mid x \in X\}|$. The topological entropy of a CA $\phi$ on $A^{\mathbb{Z}^d}$ is $h(\phi) = \lim_{r\to\infty} \lim_{n\to\infty} n^{-1} \log |\{(\phi^t(x)_{[0,r-1]^d})_{t=0}^{n-1} \mid x \in A^{\mathbb{Z}^d}\}|$.*

By classical results in symbolic dynamics (see Sections 4 and 11 of [25]), the entropy of a one-dimensional SFT is efficiently computable and its possible values are exactly the nonnegative integer multiples of logarithms of Perron numbers. The entropies of two- and higher-dimensional SFTs were characterized by Hochman in [22] as the nonnegative right-computable numbers, that is, limits of computable decreasing sequences of rational numbers. Independently in the preprint [15] using a variant of Robinson tiles and in [7] using fixed point tile sets, it was shown that they can be realized with transitive SFTs.

▶ **Theorem 11** (Gangloff & Sablik; Durand & Romashchenko). *The topological entropies of transitive 2D SFTs are exactly the nonnegative right-computable numbers.*

The topological entropies of three- and higher-dimensional CA were characterized in [20]. Guillon and Zinoviadis handled the remaining dimensions, 1 and 2, in [19]. Their proof uses the fixed point construction.

▶ **Theorem 12** (Hochman; Guillon & Zinoviadis). *The topological entropies of one-dimensional CA are exactly the nonnegative right-computable numbers, and those of two- and higher-dimensional CA are exactly the limits of increasing computable sequences of nonnegative right-computable numbers (including $\infty$).*

The high-level idea in each construction is the same. In the SFT case, given a number $\alpha \geq 0$ in the appropriate class, construct an SFT that factors onto a subshift $X \subseteq \{0,1\}^{\mathbb{Z}^d}$ where the maximal asymptotic density of 1-symbols in a configuration equals $\alpha$, and then

allow any subset of 1s to be independently replaced by 2s. In the case of cellular automata, a $d$-dimensional self-simulating CA with variable zoom factor is used to analyze a separate binary layer of the configuration, which is not modified by the CA except when erased by the error state. The CA produces a spreading error state if the local density of 1-symbols is too high, which guarantees that such patterns are transient and thus do not contribute to the entropy. This density is converted into entropy by replacing an arbitrary subset of 1s by 2s and composing the CA with a shift.

## 4.6 Expansive directions

Subshifts are *expansive* dynamical systems: a configuration $x \in X \subseteq A^{\mathbb{Z}^d}$ can be reconstructed from any family of good enough approximations of the shifts $\tau^{\vec{v}}x$ for $\vec{v} \in \mathbb{Z}^d$. Boyle and Lind studied a stronger variant called *directional expansivity*, where a certain subset of the shifts suffices to determine $x$, in [3]. For convenience, we define it only in the two-dimensional case.

▶ **Definition 13.** *Let $\vec{0} \in L \subset \mathbb{R}^2$ be a line through the origin with direction (slope) $\ell \in \dot{\mathbb{R}} = \mathbb{R} \cup \{\infty\}$. We say $\ell$ is* expansive *for a subshift $X \subset A^{\mathbb{Z}^2}$, if there exists $r \geq 0$ such that the function $x \mapsto x|_D$ is injective on $X$, where $D = (L + [-r, r]^2) \cap \mathbb{Z}^2$. Denote the nonexpansive directions by $N(X) \subseteq \dot{\mathbb{R}}$.*

For any infinite two-dimensional subshift $X$, the set $N(X)$ is nonempty and closed. Boyle, Lind and Hochman proved in [3, 21] that every nonempty closed subset $L \subseteq \dot{\mathbb{R}}$ occurs as $N(X)$ for some subshift $X$. The subshifts constructed by Boyle and Lind consist of two layers, each of which contains regularly spaced discrete approximation of lines, and it realizes every $L$ except singleton irrational directions. Hochman handled the remaining case with a subshift resembling the set of spacetime diagrams of a fixed point CA with variable zoom factor, but without the simulated Turing machines. Since $X$ is not required to be of finite type, the hierarchical structure can be enforced. Building on Hochman's proof, Guillon and Zinoviadis have characterized the sets of nonexpansive directions of SFTs in an unpublished manuscript [18], which also contains several realization results of SFTs with a unique nonexpansive direction. The results are reported in Zinoviadis's PhD dissertation [34]. In the next result, a subset $A \subseteq \dot{\mathbb{R}}$ is *effectively closed* if there exists a computable sequence $(I_n)_{n\in\mathbb{N}}$ of open intervals (which may have the form $(a, \infty] \cup (-\infty, b)$) with rational endpoints such that $\dot{\mathbb{R}} \setminus A = \bigcup_{n\in\mathbb{N}} I_n$.

▶ **Theorem 14** (Guillon & Zinoviadis)**.** *The sets $N(X)$ for two-dimensional SFTs $X$ are exactly the effectively closed subsets of $\dot{\mathbb{R}}$.*

The main technical contribution of the dissertation is a fixed point construction for reversible partial cellular automata (RPCA), which are defined by a local rule that is a partial function. Note that the fixed point CA presented in Section 3.2 is not reversible. The set of spacetime diagrams of a one-dimensional RPCA is a two-dimensional SFT for which all directions $-1 < \ell < 1$ are expansive. For an RPCA obtained by the fixed point construction with variable zoom factor, if $\prod_{k=0}^{\infty} Q_k/U_k = 0$, then every direction except $\infty$ (the vertical line) is expansive. A shift map by $D_k \in \mathbb{Z}$ cells is then applied to each level $k$ of the simulation hierarchy, which changes the direction of the unique nonexpansive line. These numbers and the simulation parameters $Q_k, U_k$ are allowed to vary within a suitable set of sequences computed by the simulated Turing machines, which provides enough control on the nonexpansive directions to realize any effectively closed set.

## References


1 Nathalie Aubrun and Mathieu Sablik. Simulation of effective subshifts by two-dimensional subshifts of finite type. *Acta Applicandae Mathematicae*, 126:35–63, 2013. doi:10.1007/s10440-013-9808-5.

2 Robert Berger. The undecidability of the domino problem. *Memoirs of the American Mathematical Society*, 66, 1966. 72 pages.

3 Mike Boyle and Douglas Lind. Expansive subdynamics. *Transactions of the American Mathematical Society*, 349(1):55–102, 1997. doi:10.1090/S0002-9947-97-01634-6.

4 Ilir Çapuni and Péter Gács. A Turing machine resisting isolated bursts of faults. *Chicago Journal of Theoretical Computer Science*, 2013(3), 2013. doi:10.4086/cjtcs.2013.003.

5 M. Delorme, J. Mazoyer, N. Ollinger, and G. Theyssier. Bulking I: an abstract theory of bulking. *Theoretical Computer Science*, 412(30):3866–3880, 2011. doi:10.1016/j.tcs.2011.02.023.

6 Bruno Durand, Leonid A. Levin, and Alexander Shen. Complex tilings. *The Journal of Symbolic Logic*, 73(2):593–613, 2008. doi:10.2178/jsl/1208359062.

7 Bruno Durand and Andrei Romashchenko. The expressiveness of quasiperiodic and minimal shifts of finite type. *Ergodic Theory and Dynamical Systems*, 41(4):1086–1138, 2021. doi:10.1017/etds.2019.112.

8 Bruno Durand, Andrei Romashchenko, and Alexander Shen. Fixed point and aperiodic tilings. In *Developments in language theory*, volume 5257 of *Lecture Notes in Computer Science*, pages 276–288. Springer, Berlin, 2008. doi:10.1007/978-3-540-85780-8_22.

9 Bruno Durand, Andrei Romashchenko, and Alexander Shen. Fixed point theorem and aperiodic tilings. *Bulletin of the European Association for Theoretical Computer Science. EATCS*, 97:126–136, 2009.

10 Bruno Durand, Andrei Romashchenko, and Alexander Shen. High complexity tilings with sparse errors. In *Automata, languages and programming. Part I*, volume 5555 of *Lecture Notes in Computer Science*, pages 403–414. Springer, Berlin, 2009. doi:10.1007/978-3-642-02927-1_34.

11 Bruno Durand, Andrei Romashchenko, and Alexander Shen. Effective closed subshifts in 1D can be implemented in 2D. In *Fields of logic and computation*, volume 6300 of *Lecture Notes in Computer Science*, pages 208–226. Springer, Berlin, 2010. doi:10.1007/978-3-642-15025-8_12.

12 Bruno Durand, Andrei Romashchenko, and Alexander Shen. Fixed-point tile sets and their applications. *Journal of Computer and System Sciences*, 78(3):731–764, 2012. doi:10.1016/j.jcss.2011.11.001.

13 Péter Gács. Reliable computation with cellular automata. *Journal of Computer and System Sciences*, 32(1):15–78, 1986. doi:10.1016/0022-0000(86)90002-4.

14 Péter Gács. Reliable cellular automata with self-organization. *Journal of Statistical Physics*, 103(1-2):45–267, 2001. doi:10.1023/A:1004823720305.

15 Silvère Gangloff and Mathieu Sablik. Quantified block gluing, aperiodicity and entropy of multidimensional SFT. *ArXiv e-prints*, 2017. arXiv:1706.01627.

16 Lawrence F. Gray. A reader's guide to Gacs's "positive rates" paper. *Journal of Statistical Physics*, 103(1-2):1–44, 2001. doi:10.1023/A:1004824203467.

17 Pierre Guillon and Gaétan Richard. Nilpotency and limit sets of cellular automata. In *Mathematical foundations of computer science 2008*, volume 5162 of *Lecture Notes in Comput. Sci.*, pages 375–386. Springer, Berlin, 2008. doi:10.1007/978-3-540-85238-4_30.

18 Pierre Guillon and Charalampos Zinoviadis. Unpublished manuscript. Work in progress.

19 Pierre Guillon and Charalampos Zinoviadis. Densities and entropies in cellular automata. In S. Barry Cooper, Anuj Dawar, and Benedict Löwe, editors, *How the world computes*, pages 253–263. Springer, 2013. doi:10.1007/978-3-642-30870-3_26.

20 Michael Hochman. On the dynamics and recursive properties of multidimensional symbolic systems. *Inventiones Mathematicae*, 176(1):131–167, 2009. doi:10.1007/s00222-008-0161-7.

21 Michael Hochman. Non-expansive directions for $\mathbb{Z}^2$ actions. *Ergodic Theory and Dynamical Systems*, 31(1):91–112, 2011. doi:10.1017/S0143385709001084.

22 Michael Hochman and Tom Meyerovitch. A characterization of the entropies of multidimensional shifts of finite type. *Annals of Mathematics. Second Series*, 171(3):2011–2038, 2010. doi:10.4007/annals.2010.171.2011.

23 Jarkko Kari. Theory of cellular automata: a survey. *Theoretical Computer Science*, 334(1-3):3–33, 2005. doi:10.1016/j.tcs.2004.11.021.

24 G. L. Kurdyumov. An example of a nonergodic one-dimensional homogeneous random medium with positive transition probabilities. *Doklady Akademii Nauk SSSR*, 238(6):1287–1290, 1978. URL: http://mi.mathnet.ru/eng/dan41542.

25 Douglas Lind and Brian Marcus. *An introduction to symbolic dynamics and coding*. Cambridge University Press, Cambridge, 1995. doi:10.1017/CBO9780511626302.

26 Jean Mairesse and Irène Marcovici. Around probabilistic cellular automata. *Theoretical Computer Science*, 559:42–72, 2014. doi:10.1016/j.tcs.2014.09.009.

27 Shahar Mozes. Tilings, substitution systems and dynamical systems generated by them. *Journal d'Analyse Mathématique*, 53:139–186, 1989. doi:10.1007/BF02793412.

28 Raphael M. Robinson. Undecidability and nonperiodicity for tilings of the plane. *Inventiones Mathematicae*, 12:177–209, 1971. doi:10.1007/BF01418780.

29 Ville Salo. On nilpotency and asymptotic nilpotency of cellular automata. In Enrico Formenti, editor, Proceedings 18th international workshop on *Cellular Automata and Discrete Complex Systems* and 3rd international symposium *Journées Automates Cellulaires,* La Marana, Corsica, September 19-21, 2012, volume 90 of *Electronic Proceedings in Theoretical Computer Science*, pages 86–96. Open Publishing Association, 2012. doi:10.4204/EPTCS.90.7.

30 Ville Salo and Ilkka Törmä. Nilpotent endomorphisms of expansive group actions. *International Journal of Algebra and Computation*, 2021. Published online. doi:10.1142/S021819672150020X.

31 Ilkka Törmä. A uniquely ergodic cellular automaton. *Journal of Computer and System Sciences*, 81(2):415–442, 2015. doi:10.1016/j.jcss.2014.10.001.

32 Hao Wang. Proving theorems by pattern recognition II. *Bell System Technical Journal*, 40:1–42, 1961. doi:10.1002/j.1538-7305.1961.tb03975.x.

33 Linda Brown Westrick. Seas of squares with sizes from a $\Pi^0_1$ set. *Israel Journal of Mathematics*, 222(1):431–462, 2017. doi:10.1007/s11856-017-1596-6.

34 Charalampos Zinoviadis. *Hierarchy and Expansiveness in Two-Dimensional Subshifts of Finite Type*. PhD thesis, University of Turku, 2016. URL: http://urn.fi/URN:ISBN:%20978-952-12-3337-1.

# Dynamical Algebraic Combinatorics, Asynchronous Cellular Automata, and Toggling Independent Sets

**Laurent David**
University of Texas, Dallas, TX, USA

**Colin Defant**
Princeton University, NJ, USA

**Michael Joseph**
Dalton State College, GA, USA

**Matthew Macauley**
Clemson University, SC, USA

**Alex McDonough**
Brown University, Providence, RI, USA

## Abstract

Though iterated maps and dynamical systems are not new to combinatorics, they have enjoyed a renewed prominence over the past decade through the elevation of the subfield that has become known as *dynamical algebraic combinatorics*. Some of the problems that have gained popularity can also be cast and analyzed as finite asynchronous cellular automata (CA). However, these two fields are fairly separate, and while there are some individuals who work in both, that is the exception rather than the norm. In this article, we will describe our ongoing work on toggling independent sets on graphs. This will be preceded by an overview of how this project arose from new combinatorial problems involving homomesy, toggling, and resonance. Though the techniques that we explore are directly applicable to ECA rule 1, many of them can be generalized to other cellular automata. Moreover, some of the ideas that we borrow from cellular automata can be adapted to problems in dynamical algebraic combinatorics. It is our hope that this article will inspire new problems in both fields and connections between them.

**2012 ACM Subject Classification** Mathematics of computing → Combinatoric problems; Theory of computation → Formal languages and automata theory; Hardware → Cellular neural networks

**Keywords and phrases** Asynchronous cellular automata, Covering space, Coxeter element, Dynamical algebraic combinatorics, Group action, Homomesy, Independent set, Resonance, Toggling, Toric equivalence

**Digital Object Identifier** 10.4230/OASIcs.AUTOMATA.2021.5

**Funding** *Colin Defant*: National Science Foundation Graduate Research Fellowship (grant no. DGE-1656466); Fannie and John Hertz Foundation Fellowship.
*Matthew Macauley*: Simons Foundation Collaboration Grant #358242.

**Acknowledgements** The authors would like to thank the Banff International Research Station, and the organizers of the Dynamical Algebraic Combinatorics Workshop, held virtually in October 2020.

## 1 Dynamical algebraic combinatorics

The subfield of dynamical algebraic combinatorics covers a wide swath of problems involving actions on combinatorial objects. Of particular interest is the study of the orbit structure, such as their sizes, and the variation of combinatorial statistics within or across orbits. One such example is the concept of *homomesy*, which is a statistic whose average value is constant over all orbits. Another popular new idea in dynamical algebraic combinatorics involves breaking up global maps into compositions of local involutions called *toggles*. These generate


so-called *toggle groups*, which can be useful in explaining certain structural properties of the orbits, such as the phenomenon of *resonance*. In this paper, we will show how, in some settings, these toggles can be modeled as an asynchronous cellular automaton (CA). For two fun and easy-to-read articles on dynamical algebraic combinatorics, homomesy, and toggling, containing many examples and colorful pictures, we will refer the reader to surveys by Roby [20] and Striker [24]. We will begin this paper with a crash course on homomesy and toggling – just enough to motivate the idea of why one should be interested in toggling independent sets, and then we will show how this can be realized with ECA rule 1. We will discuss our techniques from combinatorics, some of which can be extended to study other CAs, and conversely, how ideas from CAs can be used to pose and explore new questions in combinatorics. However, before we will do any of that, we will begin with a few concepts from the theory of Coxeter groups, which underlies both subjects.

## 1.1 Some basic Coxeter theory

The toggling operations that we will introduce in this section are involutions, and so the groups generated by these toggles will always be quotients of Coxeter groups [2]. A *Coxeter system* $(W, S)$ consists of a group $W$ generated by a set $S = \{s_1, \ldots, s_n\}$ with presentation

$$W = \left\langle s_1, \ldots, s_n \mid s_i^2 = 1,\ (s_i s_j)^{m_{ij}} = 1 \text{ for } i \neq j \right\rangle,$$

where $m_{ij} = |s_i s_j| \in \{2, 3, \ldots\} \cup \{\infty\}$. We can encode this by a **Coxeter diagram** $\Gamma(W, S)$, which is a graph with vertex set $S$ and an edge $\{s_i, s_j\}$ for each non-commuting pair, labeled with $m_{ij}$.[1] We will refer to the unlabeled version as the **Coxeter graph**, $\Gamma$. A **Coxeter element** of $W$ is the product of the generators in some order, i.e.,

$$c_\pi := s_{\pi_1} s_{\pi_2} \cdots s_{\pi_n} \in W,$$

for some permutation $\pi = \pi_1 \cdots \pi_n \in S_n$. We denote the set of Coxeter elements by $\mathrm{C}(W, S)$. There is an obvious bijection between Coxeter elements and the set $\mathrm{Acyc}(\Gamma)$ of acyclic orientations, where we orient the edge $\{s_i, s_j\}$ as $s_i \to s_j$ if $s_i$ comes before $s_j$ in $c_\pi$. Conversely, every acyclic orientation of $\Gamma$ defines a partial order on $S$, and the Coxeter element arises from any of its linear extensions.

It is well known that two Coxeter elements $c, c' \in \mathrm{C}(W, S)$ are conjugate if and only their corresponding acyclic orientations are **torically equivalent**, which means that they differ by a sequence of source-to-sink conversions [8]. In [18], Pretzel showed that a complete invariant for this is the **circulation** $\nu$ around each cycle, which is the number of "forward" edges minus the number of "backward" edges.[2]

Figure 1 shows five acyclic orientations of a Coxeter graph. The circulation around the unique cycle of the first three orientations is $\nu = 0$. For the fourth, $\nu = 2$, and for the fifth, $\nu = -2$. This means that the first three orientations represent conjugate Coxeter elements, while the other two fall into two different conjugacy classes.

## 1.2 Homomesy, rowmotion, promotion, and toggles

Consider a set $X$ of combinatorial objects. There are a surprising number of group actions on $X$ where some statistic $X \to \mathbb{R}$, though not constant, has a constant average of $c$ over all orbits. This is called a **homomesy**, and such a statistic is said to be **homomesic**, or **$c$-mesic**.

[1] Note that $s_i$ and $s_j$ commute if and only if $m_{ij} = 2$.

[2] If the graph is planar, then we can take "forward" to be clockwise and "backward" to be counterclockwise. It is elementary to extend this notion for non-planar graphs, but we do not need to do that here.

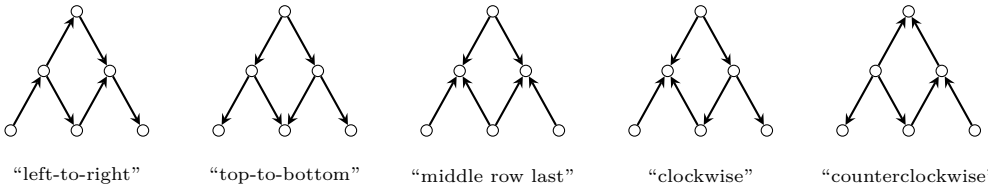



**Figure 1** Five acyclic orientations of a Coxeter graph $\Gamma$. The first three describe conjugate Coxeter elements, because their circulations around the cycle is $\nu = 0$. The last two have circulations of $\nu = 2$ and $\nu = -2$, respectively.

In many cases, the analysis into why such phenomena arise uncovers the often unexpected fact that the actions can be broken up into a sequence of involutions called *toggles*.

To introduce this idea, let $\mathcal{P}$ be a poset over $[n] = \{1, \ldots, n\}$, and let $J(\mathcal{P})$ be its set of **order ideals** (i.e., $I \subseteq \mathcal{P}$ such that if $y \in I$ and $x \leq y$, then $x \in I$). The operation of **rowmotion**, introduced in 1995 by Cameron and Fon-Der-Flaass [3], is a bijection $J(\mathcal{P}) \to J(\mathcal{P})$ defined by sending an ideal $I$ to the ideal generated by the antichain of minimal elements of $\mathcal{P} \setminus I$. An example of rowmotion on the root poset of type $A_3$ is shown in Figure 2. Elements in the ideals are denoted by solid black dots. Notice that for this example, there are three orbits of sizes 8, 4, and 2.

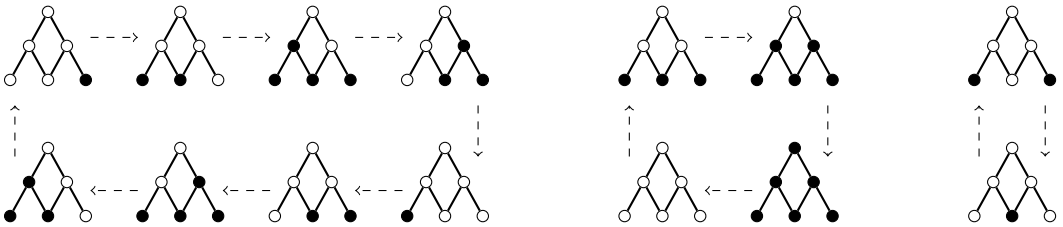


**Figure 2** Rowmotion maps an order ideal to the ideal generated by the antichain of minimal elements of its complement. This can also be realized by "toggling" elements in/out (when possible) from top-to-bottom, i.e., along any linear extension of the second acyclic orientation from Figure 1. The number of maximal elements is *homomesic* because its average is 3/2 on all orbits.

A **statistic** on order ideals is a function $J(\mathcal{P}) \to \mathbb{R}$. Three examples, which we will refer back to, include the cardinality of the ideal, the number of minimal elements, and the number of maximal elements. Notice how over the large orbit in Figure 2, the average cardinality is 5/2, the average number of minimal elements is 2, and the average number of maximal elements is 3/2. On the medium orbit, these averages are 7/2, 9/4, and 3/2, respectively. Finally, on the small orbit, they are all 3/2. Consequently, the statistic that counts the number of maximal elements is $\frac{3}{2}$-mesic.

Though it appears infeasible to analyze rowmotion systematically on arbitrary posets, it has some surprising properties on certain families. One such example is the family of *rc-posets* [26], which are characterized by having well-defined notions of "rows" and "columns," such as in Figure 2. For example, if $\mathbf{a}$ and $\mathbf{b}$ are chains of size $a$ and $b$, respectively, then rowmotion on their product $\mathcal{P} = \mathbf{a} \times \mathbf{b}$ has order $a + b$ and exhibits the cyclic sieving phenomenon [19]. These properties arise because rowmotion can be decomposed as a product of $n$ involutions, called *toggles*, one for each $k \in \mathcal{P}$. Loosely speaking, given an ideal $I$ and an element $k \in \mathcal{P}$ of a poset, toggling at $k$ attempts to add/remove $k$ to/from $I$. More precisely, **toggling** $I$ **at** $k$ means:

1. if $k \notin I$ and $I \cup \{k\} \in J(\mathcal{P})$, then add it;
2. if $k \in I$ and $I \setminus \{k\} \in J(\mathcal{P})$, then remove it;
3. otherwise, do nothing.

Toggles corresponding to elements $k, \ell$ commute when neither $k$ nor $\ell$ covers the other. Thus, in an $rc$-poset such as in Figure 2, the toggles of any two elements in the same horizontal row commute, as do the toggles of any two elements in the same vertical column. Therefore, it is well-defined to speak of toggling elements from top-to-bottom, from left-to-right, etc.

Since each individual toggle is a bijection on $J(\mathcal{P})$, the set of toggles forms a group that we call the **toggle group**, or more specifically, the *order ideal toggle group* $\mathrm{Tog}(J(\mathcal{P})) = \langle t_1, \ldots, t_n \rangle$ . Clearly, each generating toggle is an involution, and so $\mathrm{Tog}(J(\mathcal{P}))$ is the quotient of a Coxeter group. Following Coxeter theory, we will define a **Coxeter element** to be the product of all toggles of $\mathcal{P}$ in some order. The name "rowmotion" is partially inspired by the fact that it is one of these Coxeter elements, which we will denote as $\mathsf{Row}$.

▶ **Theorem 1.** *If $\mathcal{P}$ is a ranked poset, then rowmotion is the Coxeter element $\mathsf{Row} \in \mathrm{Tog}(J(\mathcal{P}))$ defined by toggling across rows, from top-to-bottom.*

One can also ask what is the result of "toggling by columns". Doing so from left-to-right is called *promotion*, denoted $\mathsf{Pro} \in \mathrm{Tog}(J(\mathcal{P}))$, because for certain posets, it agrees with a well-known operation with the same name, first introduced in 1972 by Schützenberger [23]. Promotion was originally defined as an action on the set $\mathcal{L}(\mathcal{P})$ of linear extensions of a poset. Striker and Williams showed that promotion on two-row rectangular tableaux $\mathrm{SYT}(2 \times b)$ is equivalent to promotion on the positive root poset of Coxeter type $A_{b-1}$ [26]. The action of promotion on the root poset of type $A_3$ is shown in Figure 3.

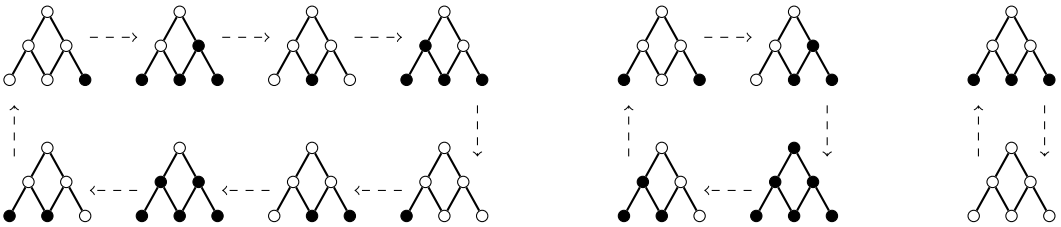


**Figure 3** Promotion can be realized by toggling from left-to-right, i.e., any linear extension of the first acyclic orientation in Figure 1. The average cardinality of the ideals, and of the minimal elements, is the same as for rowmotion, but the statistic that counts the number of maximal elements is no longer homomesic. In particular, its average is 13/8, 5/4, and 3/2 across the three orbits.

One immediate observation from Figures 2 and 3 is that promotion and rowmotion have the same orbit structure. This is a simple consequence from Coxeter theory – since the acyclic orientation defined by toggling top-to-bottom has the same circulation as the one defined by toggling left-to-right, they are torically equivalent. Therefore, the corresponding Coxeter elements $\mathsf{Row}$ and $\mathsf{Pro}$ are conjugate in $\mathrm{Tog}(J(\mathcal{P}))$. However, not all statistics are preserved by conjugation. Observe that the average order ideal cardinality, and the average number of minimal elements, are the same for promotion and rowmotion. However, the number of maximal elements is no longer homomesic: it is 13/8 for the large orbit, 5/4 for the medium orbit, and 3/2 for the small orbit. It is easy to check from Figures 2 and 3 that none of these statistics are preserved elementwise by conjugation from rowmotion to promotion, but the average number of elements in each orbit is preserved.

Another fundamental feature of an action on a set of combinatorial objects is the collection of orbit sizes. Though each of these has to divide the size of the toggle group, in practice, such groups are often large, and this does not provide meaningful information. In a number of cases, computational experiments are used to answer this question for small $n$. It has often been observed that most, but not necessarily all, of the orbits in an action are divisible

by some integer $\omega > 1$. Such actions are said to **resonate** with $\omega$, though for years, people did not know exactly how to precisely define this, because it involved "most," rather than "all" orbits. For example, on the poset $\mathcal{P} = \mathbf{a} \times \mathbf{b} \times \mathbf{c}$, a product of three chains, the sizes of most orbits under rowmotion are multiples of $a+b+c-1$. The sizes of all 21156 rowmotion orbits of $\mathcal{P} = \mathbf{4} \times \mathbf{4} \times \mathbf{4}$ are multiples of 11: six have size 33 and the others all have size 11. However, for the rowmotion orbits of $\mathcal{P} = \mathbf{4} \times \mathbf{4} \times \mathbf{5}$, most of the orbits have sizes divisible by 12 (136822 of size 12, 4 of size 24, 100 of size 36), but there are also 6 orbits of size 2, 8 of size 3, and 162 of size 6. In 2015, the concept of *resonance* was formally defined, involving commutative diagrams of actions of cyclic groups [6].

## 1.3 Generalized toggling

The idea of toggling an order ideal $I$ at an element $k$ by attempting to add/remove it is easy to generalize to other combinatorial objects [25]. Consider a finite set $X$, such as a graph or a poset, and a collection $\mathcal{L} \subseteq 2^X$ of subsets that have a particular property (such as "is an order ideal"). For each $k \in X$ and subset $E \in \mathcal{L}$, we say that **toggling** $E$ **at** $k$ is the function

$$t_k \colon \mathcal{L} \longrightarrow \mathcal{L}, \qquad t_k(E) = \begin{cases} E \cup \{k\} & k \notin E \text{ and } E \cup \{k\} \in \mathcal{L} \\ E \setminus \{k\} & k \in E \text{ and } E \setminus \{k\} \in \mathcal{L} \\ E & \text{otherwise.} \end{cases}$$

We sometimes speak of the first case above as "*toggling in* $k$" and the second case as "*toggling out* $k$." It should be clear that $t_k$ is either the identity or an involution.

For some examples of generalized toggling, we can always toggle out, but not always toggle in. For example, if we toggle the set $\mathcal{L} = \mathcal{A}(\mathcal{P})$ of antichains of a poset $\mathcal{P}$, then $t_k(A) = A \setminus \{k\}$ whenever $k \in A$, because removing any element from an antichain results in another antichain. This is closely related to order ideal toggling, because every order ideal is uniquely determined by an antichain – the minimal elements of its complement. In particular, the toggle groups are isomorphic, despite the individual toggles having fundamentally different properties [9]. In contrast, there are other settings where we can always toggle in, but not necessarily toggle out. Toggling vertex covers, edge covers, or dominating sets of a graph are such examples.

Sometimes, there are sets of objects that do not seem to be "togglable" upon first glance, but admit a nice toggle action if one views them the right way. For example, a **noncrossing partition** of $[n]$ is a partition such that if the numbers $1, \ldots, n$ are arranged cyclically, then the convex hulls of the blocks are disjoint. An alternate way to represent a noncrossing partition is with an **arc diagram**: draw the numbers $1, \ldots n$ in a line, and draw an arc between all consecutive vertices in the same block. By construction, every collection of arcs on $[n]$ represents a noncrossing partition if it does not contain any of the three motifs shown in Figure 4.


**Figure 4** Disallowed pairs of arcs in a noncrossing partition: crossing, left-nesting, and right-nesting, respectively.

Let $X = \{\{i,j\} \mid 1 \leq i < j \leq n\}$ be the set of arcs on $[n]$, and $\mathrm{NC}(n) \subseteq 2^X$ the set of noncrossing partitions. It should now be easy to see how to toggle a noncrossing partition $\pi \in \mathrm{NC}(n)$. Specifically, for each arc $\{i,j\} \in X$, the toggle $t_{i,j}$ removes it from $\pi$ if it already

contains it. Otherwise, $t_{i,j}$ adds the arc, as long as $\pi \cup \{\{i,j\}\}$ does not contain one of the forbidden motifs from Figure 4. Notice that arcs can always be toggled out, but not necessarily in. Figure 5 shows an example of toggling a noncrossing partition of size $n = 12$, both as arc diagrams and as convex hulls. Computational experiments originally suggested the surprising fact that under any Coxeter element, the arc count statistic is $\frac{n-1}{2}$-mesic, and this was proven in [7].

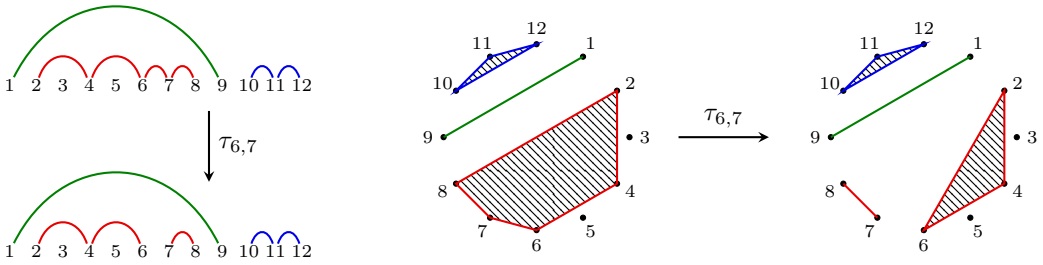



**Figure 5** An example of toggling a noncrossing partition of $[n] = [12]$.

The Coxeter graph of $\mathrm{Tog}(\mathrm{NC}(n))$ can be constructed by putting the toggle $t_{i,j}$ in the $(i,j)$ position of a grid, arranged in the upper-triangular entries of a matrix. In doing so, each column and each row is a complete graph (i.e., no two toggles in the same row or column commute), and there are some other edges as well, the details of which are not important here, other than the fact that they all have a "negative slope." An example of this graph for $n = 5$ is shown in Figure 6. One key observation is that the Coxeter element by rows (from top-to-bottom) and the Coxeter element by columns (from left-to-right) both arise from the same acyclic orientation, constructed by orienting all edges in the "south, east, or southeast directions." In other words, toggling by rows is the same element in the toggle group as toggling by columns. It turns out that this element is the inverse of the well-studied *Kreweras complement* $\kappa$ on noncrossing partitions [11], an operation that has the curious property that $\kappa^2$ is a rotation of the "convex hull" diagram by $2\pi/n$. In [1], this map, along with rowmotion, was used to construct a uniform bijection between noncrossing and nonnesting partitions in Weyl groups of all classical types.

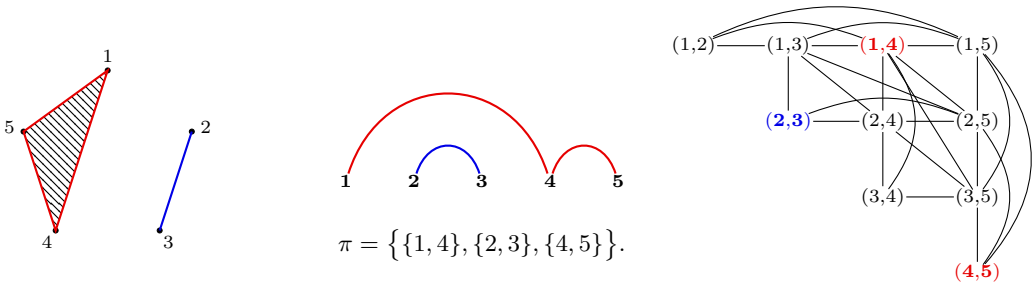



**Figure 6** There is a bijection between noncrossing partitions on $[n]$ and independent sets of the Coxeter diagram of the NC toggle group. An example of this for $n = 5$ is shown.

The arcs in a noncrossing partition form an independent set of the Coxeter graph, and every such independent set corresponds to a noncrossing partition. In other words, toggling noncrossing partitions is just a special case of a more general construction: toggling independent sets in a graph. However, the Coxeter graphs of the NC toggle group are quite

complicated. The third author and Roby studied toggling independent sets of the path graph in [10], and the toggle groups were recently shown to be symmetric groups [17]. In an ongoing project [4], the authors of this paper are studying toggling independent sets of the cycle graph. Early computational experiments, motivated by the search for examples of homomesy and resonance, brought other curious properties to light. For example, if one records the frequency with which each vertex is toggled in over time, this cyclic vector always has odd period under the update order $\pi = 1, \ldots, n$, but not necessarily under other update orders. There are interesting observations to be made about the period and the sizes of the orbits, as well as how this depends on the update order and on $n$. In the next section, we will show how this framework can be viewed as an asynchronous CA. We will also describe the analytical tools and techniques we developed, inspired by ideas from dynamical algebraic combinatorics, that we think are mostly new in the field of cellular automata. Additionally, we will discuss ideas from cellular automata that we are bringing back to the field of dynamical algebraic combinatorics.

## 2 Toggling independents sets in a cellular automata framework

### 2.1 Asynchronous cellular automata

A *cellular automaton* (CA) is a discrete dynamical system over a regular grid. A one-dimensional CA is either over an infinite path or a cycle graph $\mathcal{C}_n$. We will assume that $\mathcal{C}_n$ has vertex set $v(\mathcal{C}_n) = [n]$, sometimes called **cells**, and take the indices modulo $n$, when appropriate. Vertex $i$ has a Boolean state $x_i \in \mathbb{F}_2 = \{0, 1\}$ and a function that updates this state based on the states of itself and its neighbors. We encode each function $f_i \colon \mathbb{F}_2^3 \to \mathbb{F}_2$ with a number between 0 and 255, based on the binary string $a_7 a_6 \cdots a_1 a_0$, defined from its truth table, as follows:

| $x_{i-1}x_ix_{i+1}$ | 111 | 110 | 101 | 100 | 011 | 010 | 001 | 000 |
|---|---|---|---|---|---|---|---|---|
| $f_i(x_{i-1}, x_i, x_{i+1})$ | $a_7$ | $a_6$ | $a_5$ | $a_4$ | $a_3$ | $a_2$ | $a_1$ | $a_0$ |

These functions are called **elementary cellular automata (ECA) rules**, and have been well-studied since CAs gained popularity in the 1980s. In a classical CA, the individual functions are updated synchronously to define the dynamical system map $f \colon \mathbb{F}_2^n \to \mathbb{F}_2^n$, which is iterated to generate the global dynamics. In this section, we will consider a common asynchronous framework, where the dynamical system map is defined by updating each function in some predetermined order, sometimes called a *fixed sweep* [22].

Fix an ECA rule, and for each vertex function $f_i \colon \mathbb{F}_2^3 \to \mathbb{F}_2$, define the function

$$F_i \colon \mathbb{F}_2^n \longrightarrow \mathbb{F}_2^n, \qquad F_i \colon (x_1, \ldots, x_i, \ldots, x_n) \longmapsto \big(x_1, \ldots, f_i(x_{i-1}, x_i, x_{i+1}), \ldots, x_n\big), \tag{1}$$

which simply updates the global state vector $(x_1, \ldots, x_n)$ at only the $i^{\text{th}}$ position. Let $\pi = \pi_1 \cdots \pi_n \in S_n$ be a permutation of the vertices of $\mathcal{C}_n$, written as a word in $v(\mathcal{C}_n)$. The fixed sweep **asynchronous cellular automaton (ACA)** map with respect to $\pi$ is

$$F_\pi \colon \mathbb{F}_2^n \longrightarrow \mathbb{F}_2^n, \qquad F_\pi = F_{\pi_n} \circ \cdots \circ F_{\pi_1}.$$

A point $x \in \mathbb{F}_2^n$ is *periodic* if $F_\pi^k(x) = x$ for some $k \geq 1$, and *transient* otherwise. The fixed points are those for which $F_\pi(x) = x$, and they do not depend on $\pi$. The periodic points are partitioned into **periodic orbits**. An ACA is a special case of a *sequential dynamical system* (SDS), which is defined over arbitrary graphs and with update functions that need not be the same at every vertex [16].

The dynamics of an ACA depends on the update order $\pi = \pi_1 \cdots \pi_n$, which we can encode as an acyclic orientation $O_\pi$ of $\mathcal{C}_n$, just as we did for Coxeter elements. This defines a map

$$S_n \longrightarrow \mathrm{Acyc}(\mathcal{C}_n), \qquad \pi \longmapsto O_\pi,$$

and it is easy to see that if $O_\pi = O_{\pi'}$, then $F_\pi = F_{\pi'}$ as ACA maps, for any fixed ECA rule. Moreover, if $O_\pi$ and $O_{\pi'}$ are torically equivalent, then the ACA maps $F_\pi$ and $F_{\pi'}$ are topologically conjugate when restricted to their respective sets of periodic points [14].

## 2.2 Togglable ACAs

An ACA is **togglable** if its set of periodic points does not depend on the update order $\pi \in S_n$. When this happens, each function $F_i$ restricted to the periodic points is a bijection, and $F_i^2 = 1$, since it either fixes or flips each $i^{\text{th}}$ bit. In other words, if ECA rule $k$ is togglable, then we have a natural toggle action on the set $\mathrm{Per}_n^{(k)}$ of periodic points, and we denote the toggle group by

$$\mathrm{Tog}_n^{(k)} = \langle F_1, \ldots, F_n \rangle .$$

It is known that 104 of the 256 ECA rules are togglable [13], and their toggle groups have been classified [12]. Those two papers predate the notion of generalized toggling, and so in them, the property of being togglable is called *order independent*, and the toggle groups are called *dynamics groups*. It is easy to see that $\mathrm{Tog}_n^{(k)}$ is trivial if and only if all periodic points are fixed points.

Every ECA rule is equivalent to the ECA rules formed by "mirror reflection" (swapping $x_{i-1}$ with $x_{i+1}$), "logical complement" (swapping 0 with 1), and doing both operations. Some rules are their own reflection, own complement, or both. As such, the 256 ECA rules are partitioned into 88 equivalence classes, of sizes 1, 2, and 4. Of these, exactly 41 classes are togglable, but some are for obvious reasons: 26 have trivial toggle groups (i.e., only fixed points), 9 are reversible (i.e., the maps $\mathbb{F}_2^n \to \mathbb{F}_2^n$ in Eq. (1) are invertible), and 1 is both. Some of the invertible rules have interesting toggle groups, and these are described in Section 5 of [12]. The conjectured group for ECA rules 54 and 57 (also conjectured independently in [27]) was recently proven by Salo [21]. Two of the invertible rules: the parity function and its complement (rules 150 and 105), each of which is in a size-1 equivalence class, were recently studied in the context of toggling by the second author [5].

In this paper, we are doing the opposite: taking a problem from combinatorics, toggling independent sets in $\mathcal{C}_n$, and studying it in an ACA framework. Specifically, an **independent set** of $\mathcal{C}_n$ is a length-$n$ binary string with no consecutive 1's, including wrapping around the end. Let $\mathcal{I}_n$ be the set of independent sets of $\mathcal{C}_n$. Toggling in $\mathcal{I}_n$ can be realized by one of several rules, such as ECA rule 1, or the invertible rule 201:

| $x_{i-1}x_ix_{i+1}$ | 111 | 110 | 101 | 100 | 011 | 010 | 001 | 000 |
|---|---|---|---|---|---|---|---|---|
| $f_i^{(1)}(x_{i-1}, x_i, x_{i+1})$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| $f_i^{(201)}(x_{i-1}, x_i, x_{i+1})$ | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |

The only differences between these two rules are on the three inputs that do not arise for independent sets: 111, 110, and 011. We are primarily interested in studying the dynamics of ECA rule 1 and $\mathrm{Tog}_n^{(1)}$, because $\mathrm{Per}_n^{(1)} = \mathcal{I}_n$, whereas $\mathrm{Per}_n^{(201)} = \mathbb{F}_2^n$. However, since rule 201 is invertible, we have a surjective homomorphism $\mathrm{Tog}_n^{(201)} \to \mathrm{Tog}_n^{(1)}$ between toggle groups.

The third author and Roby studied toggling independent sets of path graphs [10]. One simplification of this framework is that all acyclic orientations of a tree are torically equivalent, and so all Coxeter elements are conjugate. In contrast, over the cycle graph $\mathcal{C}_n$, there are $n-1$ toric equivalence classes, distinguished by their circulations $\nu = n-2, n-4, \ldots, -(n-4), -(n-2)$. An example of this for $n = 6$ is shown in Figure 7. However, since a vertical reflection, which is an automorphism of $\mathcal{C}_n$, reverses the sign of the circulation, there are at most $\lfloor \frac{n}{2} \rfloor$ equivalence classes up to dynamical equivalence.

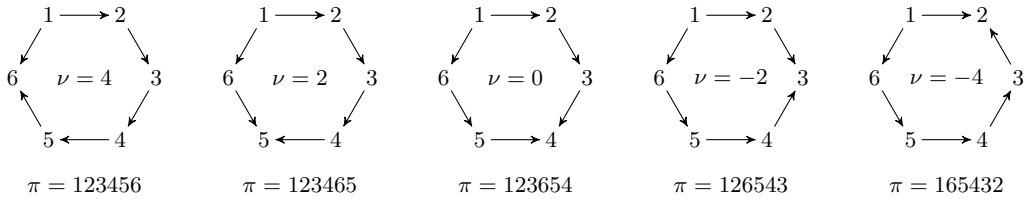



**Figure 7** Representatives of the five torically non-equivalent Coxeter elements of $\mathcal{C}_6$ and their circulations $\nu$, which fall into only three classes under the action of $\mathrm{Aut}(\mathcal{C}_6)$.

As a consequence, to understand toggling independent sets over $\mathcal{C}_n$, we only need to consider $\lfloor \frac{n}{2} \rfloor$ update orders. Specifically, we may choose those of the form

$$\pi = \quad 12\cdots k\, n(n-1)\cdots(k+1), \quad \text{for } k = \left\lceil \frac{n}{2} \right\rceil, \ldots, n-1. \tag{2}$$

Experimental evidence has shown that, for even $n$, under the **identity update order** $\pi = 12\cdots n$, the ACA map tends to have larger orbits than over other update orders. We have also observed a peculiar feature about the periodicity of the average number of 1s. For these reasons, we began our analysis using this update order, and that is what we will use for the remainder of this paper. In the next section, we will describe the methods we developed and preliminary results.

Though our focus in the remainder of this paper is on ECA rule 1 under $\pi = 12\cdots n$, we want our techniques to be adaptable to other update orders, other togglable ECA rules, and more general toggle actions. Since ECA rule 1 only has one 1 in its truth table, the occurrences of 1s in its periodic orbits are sparse. As such, to understand the dynamics, it helps to look at the pattern of the 1s. If we have a rule that has mostly 1s, then its complement is an equivalent rule that has mostly 0s. Thus, we can always assume, without loss of generality, that we are studying a rule that has at least as many 0s as 1s in its periodic orbits. We will refer to such rules as being **sparse**.

## 2.3 Scrolls and ticker tapes

In this section, we will describe some tools and results for toggling independent sets of $\mathcal{C}_n$. We will view them in two formats: a bi-infinite periodic table of 0s and 1s called the *scroll*, and a bi-infinite sequence called the *ticker tape*. A function called the *successor* on the *live entries* (the set of 1s) defines equivalence classes called *snakes*. A "dual" commuting function called the *shadow* allows us to travel between snakes, and defines less visually obvious equivalence classes called *co-snakes*. These functions generate the abelian *snake group* that acts simply transitively on the set of live entries. If we quotient out by a periodic orbit, sending the infinite scroll to a finite table, then each snake (respectively, co-snake) merges into a finite *ouroboros* (respectively, *co-ouroboros*) on a torus. Topologically, this is a covering map, and it induces a homomorphism from the snake group to the *ouroboros group*, that acts simply

transitively on the live entries in the finite orbit table. Studying this action algebraically allows us to deduce properties about the structure of the dynamics. Near the end of this paper, we will formalize an abstraction of this, called *commuting complementary pairs*, which can be adapted to other update orders, other ECA rules, and even arbitrary binary tables and other combinatorial actions.

Fix an initial state $x = x^{(0)} = (x_1, \ldots, x_n) \in \mathcal{I}_n$ and update order $\pi = \pi_1 \cdots \pi_n \in S_n$. Let $x^{(i)}$ be the result of iterating $F_\pi$ exactly $i$ times from $x$. Since $F_\pi$ is bijective on $\mathcal{I}_n$, we can define this for all $i \in \mathbb{Z}$. Consider the bi-infinite table with $n$ columns, indexed by $j = 1, \ldots, n$, and rows indexed downward by $i \in \mathbb{Z}$. The $(i,j)$-entry $X_{i,j}$ is the state of cell $v_{\pi_j}$ in $x^{(i)}$. In other words, the $i^{\text{th}}$ row consists of the states of the cells at time $t = i$, in the order that they are updated. This infinite table is called the **scroll** of $x$ and $\pi$, and denoted $\mathcal{S} = (X_{i,j}) = \mathsf{Scroll}(x, \pi)$.

The global dynamics of $F_\pi$ is read off the scroll as one reads from a book: left-to-right across rows, and the rows vertically downward. This defines a bi-infinite sequence called the **ticker tape**, denoted $\mathcal{X} = (X_k) = \mathsf{Tape}(x, \pi)$. We will index the entries so $X_0 = X_{0,\pi_1}$, $X_1 = X_{0,\pi_2}$, and so on, so in general, $X_r = X_{\lfloor r/n \rfloor, \pi_{1+(r \bmod n)}}$. It is elementary to translate results in the orbit table framework to ticker tapes or vice-versa, but sometimes, one format is easier to work with that the other. Formally, the **live entries**, in both formats, are the sets

$$\mathsf{Live}(\mathcal{S}) = \big\{(i,j) \in \mathbb{Z} \times \mathbb{Z}_n \mid X_{i,j} = 1\big\}, \qquad \mathsf{Live}(\mathcal{X}) = \big\{k \in \mathbb{Z} \mid X_k = 1\big\}.$$

An example of a portion of a scroll, corresponding to a single periodic orbit, is shown twice in Figure 8. In the top one, the live entries are colored by equivalence classes called *snakes*, and in the bottom one, by *co-snakes*. We will refer back to this example when we formally define these and related terms, such as the successor, shadow, and slither.

| $x$ | $v_1$ | $v_2$ | $v_3$ | $v_4$ | $v_5$ | $v_6$ | $v_7$ | $v_8$ | $v_9$ | $v_{10}$ | $v_{11}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $x^{(0)}$ | 0 | 0 | 0 | 0 | **1** | 0 | **1** | 0 | 0 | 0 | 0 |
| $x^{(1)}$ | **1** | 0 | **1** | 0 | 0 | 0 | 0 | **1** | 0 | **1** | 0 |
| $x^{(2)}$ | 0 | 0 | 0 | **1** | 0 | **1** | 0 | 0 | 0 | 0 | **1** |
| $x^{(3)}$ | 0 | **1** | 0 | 0 | 0 | 0 | **1** | 0 | **1** | 0 | 0 |
| $x^{(4)}$ | 0 | 0 | **1** | 0 | **1** | 0 | 0 | 0 | 0 | **1** | 0 |
| $x^{(5)}$ | **1** | 0 | 0 | 0 | 0 | **1** | 0 | **1** | 0 | 0 | 0 |
| $x^{(6)}$ | 0 | **1** | 0 | **1** | 0 | 0 | 0 | 0 | **1** | 0 | **1** |

$\cdots\ 0\ 0\ \cdots$
$\cdots\ 0\ \mathbf{1}\ 0\ \mathbf{a}\ \cdots$
$\cdots\ 0\ 0\ \overline{\mathbf{a}}\ 0\ \cdots$
"*successor of* $(i,j)$"

| $x$ | $v_1$ | $v_2$ | $v_3$ | $v_4$ | $v_5$ | $v_6$ | $v_7$ | $v_8$ | $v_9$ | $v_{10}$ | $v_{11}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $x^{(0)}$ | 0 | 0 | 0 | 0 | **1** | 0 | **1** | 0 | 0 | 0 | 0 |
| $x^{(1)}$ | **1** | 0 | **1** | 0 | 0 | 0 | 0 | **1** | 0 | **1** | 0 |
| $x^{(2)}$ | 0 | 0 | 0 | **1** | 0 | **1** | 0 | 0 | 0 | 0 | **1** |
| $x^{(3)}$ | 0 | **1** | 0 | 0 | 0 | 0 | **1** | 0 | **1** | 0 | 0 |
| $x^{(4)}$ | 0 | 0 | **1** | 0 | **1** | 0 | 0 | 0 | 0 | **1** | 0 |
| $x^{(5)}$ | **1** | 0 | 0 | 0 | 0 | **1** | 0 | **1** | 0 | 0 | 0 |
| $x^{(6)}$ | 0 | **1** | 0 | **1** | 0 | 0 | 0 | 0 | **1** | 0 | **1** |

$\cdots\ 0\ 0\ \cdots$
$\cdots\ 0\ \mathbf{1}\ 0\ \cdots$
$\cdots\ 0\ 0\ \cdots$
$\cdots\ \mathbf{b}\ \overline{\mathbf{b}}\ \cdots$
"*shadow of* $(i,j)$"

**Figure 8** The scroll of ECA rule 1, with $\pi = 12\cdots n$, consists of the seven rows $x^{(0)}, \ldots, x^{(6)}$, repeated indefinitely. This is shown twice to emphasize the snakes and co-snakes, separately. The top scroll highlights the two snakes, which are generated by the successor function, and have standard slither $(D2)^3$. In the bottom scroll, the colors distinguish the six co-snakes, generated by the shadow function, which have standard co-slither $S^2$.

## 2.4 Snakes in scrolls

For the remainder of this section, we will assume that our update order is $\pi = 12\cdots n$, and we will omit this from the notation and write, e.g., $\mathsf{Scroll}(x)$ and $\mathsf{Tape}(x)$. If we take an arbitrary live entry $(i, j)$, we can draw conclusions about its surrounding entries. For example, the entries to the left and right ($X_{i,j-1}$ and $X_{i,j+1}$) are both 0, as are the entries directly above and below ($X_{i-1,j}$ and $X_{i+1,j}$). Since the scroll is just a rendering of the ticker tape, when we get to the end of a row, the next entry is the first entry of the following row. As such, we will assume that $X_{i,n+1} = X_{i+1,1}$.

It is not hard to show that for every live entry $(i, j)$, we must have $X_{i,j+2} + X_{i+1,j+1} = 1$. We will call the coordinates of whichever of these states is live the **successor** of $(i, j)$, denoted $s(i, j)$. Similarly, we also must have $X_{i+2,j-2} + X_{i+2,j-1} = 1$, and we call the coordinates of whichever of these is live the **shadow**, denoted $c(i, j)$. Iterating the successor function defines equivalence classes on $\mathsf{Live}(\mathcal{S})$ that we will call *snakes*, due to their visual appearance in the scroll, as shown in Figure 8. Iterating the shadow function defines equivalence classes that we will call *co-snakes*.

▶ **Definition 2.** *Given a live entry $(i, j)$ in a scroll, the **snake** and the **co-snake** containing it are the sets*

$$\mathsf{Snake}(i, j) = \{s^k(i, j) \mid k \in \mathbb{Z}\}, \qquad \mathsf{CoSnake}(i, j) = \{c^k(i, j) \mid k \in \mathbb{Z}\}.$$

A key property about the successor and shadow functions, illustrated by Figure 9, is that they commute.

▶ **Proposition 3.** *The successor of the shadow is the shadow of the successor. That is, for any live entry $(i, j)$, we have $s(c(i, j)) = c(s(i, j))$.*


**Figure 9** Given a live entry $X_{i,j} = 1$, the two cases shown here illustrate the property that "*the successor of the shadow is the shadow of the successor*".

We can record the "shape" of a snake by starting at any live entry $(i, j)$, iterating the successor function, and recording a 2 or a $D$ depending on whether the successor is (in ticker tape notation) $X_{i+2}$ or $X_{i+n+1}$. This defines a periodic bi-infinite sequence, and a subsequence that generates it is called a **slither**. Two slithers are equivalent if they generate the same bi-infinite sequence. We can use exponents for ease of notation, e.g., $(2D)^3 = 2D2D2D \sim D2D2D2 = (D2)^3$. Of course, these are equivalent to $2D$ and $D2$, but as we will see, the most canonical form of a slither is not necessarily a minimal periodic subsequence. Similarly, we can record the shape of a co-snake with a bi-infinite sequence called its **co-slither**. If $(i, j)$ is live, then its shadow is either $(i - 1, j + 2)$ or $(i - 2, j + 2)$. We will refer to the former as a *short step* and the latter as a *long step*, and denote these by $S$ and $L$, respectively.

▶ **Proposition 4.** *In any scroll, all snakes have a common slither, and all co-snakes have a common co-slither.*

If we start at $(i,j) \in \mathsf{Live}(\mathcal{S})$ and iterate the successor function until we return to the same co-snake, then the corresponding sequence of 2s and $D$s is called the **standard slither**. Similarly, iterating the shadow function until we return to the same snake, defines the **standard co-slither**. This allows us to prove the following.

▶ **Proposition 5.** *For any live entry $(i,j)$ in $\mathcal{S}$,*

$$s^{\#\text{co-snakes}}(i,j) = c^{\#\text{snakes}}(i,j). \tag{3}$$

Returning to our example in Figure 8, it is easy to check that the two snakes have standard slither $(D2)^3$, the six co-snakes have standard slither $S^2$, and that $s^6 = c^2$.

The successor and shadow functions generate an abelian group, which we write multiplicatively to suggest function composition, and because biologically, not all snakes are adders.

▶ **Definition 6.** *Suppose $\mathcal{S}$ has $\alpha$ snakes and $\beta$ co-snakes. The **snake group** is defined as*

$$G(\mathcal{S}) = \left\langle s, c \mid sc = cs,\ s^\beta = c^\alpha \right\rangle.$$

It is not clear *a priori* whether there are other relations between the successor and shadow functions, but the following guarantees that there are none.

▶ **Theorem 7.** *The snake group $G(\mathcal{S})$ acts simply transitively on $\mathsf{Live}(\mathcal{S})$.*

This action endows $\mathsf{Live}(\mathcal{S})$ with the structure of a Cayley diagram for $G(\mathcal{S})$. Moreover, the snakes and co-snakes are cosets of $\langle s \rangle$ and $\langle c \rangle$, respectively. The number of snakes is $[G(\mathcal{S}) : \langle c \rangle]$, the length of the standard co-slither, and the number of co-snakes is $[G(\mathcal{S}) : \langle s \rangle]$, the length of the standard slither.

## 2.5 Ouroboroi in orbit tables

Though the scroll is infinite, it will often be helpful to restrict our attention to a single orbit, and allow snakes and co-snakes to "wrap around" from bottom-to-top. We will refer to such a "circular snake" as an *ouroboros*, inspired by the ancient symbol of a snake swallowing its tail. Suppose $x$ lies in a periodic orbit $x = x^{(0)}, \ldots, x^{(m-1)}, x^{(m)} = x$ of size $m$. The $m \times n$ table with top row $x^{(0)}$ and bottom row $x^{(m-1)}$ is the **orbit table**, denoted $\mathcal{T} = \mathsf{Table}(x, \pi)$, or $\mathsf{Table}(x)$ if $\pi = 12\cdots n$ is understood.

Let $\mathsf{Live}(\mathcal{T}) \subseteq \mathbb{Z}_m \times \mathbb{Z}_n$ be the set of live entries in the orbit table, which is the image of the set of live entries of the scroll under the natural quotient map $p\colon \mathsf{Live}(\mathcal{S}) \to \mathsf{Live}(\mathcal{T})$ that reduces the first coordinate modulo $m$. The shadow and successor functions descend to bijections on $\mathsf{Live}(\mathcal{T})$ that we will call the **mod $m$ successor function** $\bar{s}$, and **mod $m$ shadow function** $\bar{c}$, respectively. This is illustrated by the following commutative diagram; there is an analogous one for $c$ and $\bar{c}$.

$$\begin{array}{ccc} \mathsf{Live}(\mathcal{S}) & \xrightarrow{s} & \mathsf{Live}(\mathcal{S}) \\ {\scriptstyle p}\downarrow & & \downarrow{\scriptstyle p} \\ \mathsf{Live}(\mathcal{T}) & \xrightarrow{\bar{s}} & \mathsf{Live}(\mathcal{T}) \end{array} \qquad \begin{array}{ccc} (a+km, b) & \xmapsto{s} & s(a+km, b) \\ {\scriptstyle p}\downarrow & & \downarrow{\scriptstyle p} \\ (a,b) & \xmapsto{\bar{s}} & \bar{s}(a,b) \end{array}$$

The functions $\bar{s}$ and $\bar{c}$ generate a finite abelian group called the **ouroboros group**, denoted $G(\mathcal{T})$. Cosets of $\langle \bar{s} \rangle$ and $\langle \bar{c} \rangle$ are called *ouroboroi* and *co-ouroboroi*, respectively.

▶ **Definition 8.** *Given a live entry $(i, j)$ in an orbit table, the* ***ouroboros*** *and* ***co-ouroboros*** *containing it are the sets*

$$\mathsf{Ouro}(i,j) = \{\bar{s}^k(i,j) \mid k \in \mathbb{Z}\}, \qquad \mathsf{CoOuro}(i,j) = \{\bar{c}^k(i,j) \mid k \in \mathbb{Z}\}.$$

The quotient map $p\colon \mathsf{Live}(\mathcal{S}) \to \mathsf{Live}(\mathcal{T})$ induces a homomorphism $p_*\colon G(\mathcal{S}) \to G(\mathcal{T})$ sending $s \mapsto \bar{s}$ and $c \mapsto \bar{c}$. The **(co-)ouroboros degree** is the number of (co-)snakes in the $p$-preimage of each (co-)ouroboros. We denote these as

$$\deg(p_*) := \frac{[G(\mathcal{S}) : \langle s \rangle]}{[G(\mathcal{T}) : \langle \bar{s} \rangle]}, \qquad \mathrm{codeg}(p_*) := \frac{[G(\mathcal{S}) : \langle c \rangle]}{[G(\mathcal{T}) : \langle \bar{c} \rangle]}.$$

Returning back to the example from Figure 8, the portions of the scrolls shown can also be considered as orbit tables. It is easy to check by applying the mod $m$ shadow function to any live entry on the bottom row that both snakes merge into a single ouroboros, hence the ouroboros degree is two. Similarly, the co-ouroboros degree is three, because the six co-snakes merge into two co-ouroboroi.

▶ **Theorem 9.** *Suppose $\mathcal{S}$ has $\alpha$ snakes, $\beta$ co-snakes, $\bar{\alpha}$ ouroboroi, $\bar{\beta}$ co-ouroboroi, and that $\tau = |\mathsf{Live}(\mathcal{T})|$. The ouroboros group has presentation*

$$G(\mathcal{T}) = \left\langle \bar{s}, \bar{c} \;\middle|\; \bar{s}\bar{c} = \bar{c}\bar{s},\ \bar{s}^{\beta} = \bar{c}^{\alpha},\ \bar{s}^{\tau/\bar{\alpha}} = \bar{c}^{\tau/\bar{\beta}} = 1 \right\rangle,$$

*and it acts simply transitively on* $\mathsf{Live}(\mathcal{T})$.

Recall that we can endow $\mathsf{Live}(\mathcal{S})$ with a natural Cayley diagram structure of the snake group. The simply transitive action of $G(\mathcal{T})$ on $\mathsf{Live}(\mathcal{T})$ gives it a Cayley diagram structure for the ouroboros group, and the quotient map $p\colon \mathsf{Live}(\mathcal{S}) \to \mathsf{Live}(\mathcal{T})$ is a covering map.

Another advantage of orbit tables is that we can easily define the **sum vector** as the $n$-tuple that records the number of live entries in each column. Early computational results suggested the curious property that the period of the sum vector, viewed as a cyclic word, is always odd.

▶ **Proposition 10.** *For update order $\pi = 12\cdots n$, the sum vector of any orbit table has odd period.*

| $x$ | $v_1$ | $v_2$ | $v_3$ | $v_4$ | $v_5$ | $v_6$ | $v_7$ | $v_8$ | $v_9$ | $v_{10}$ | $v_{11}$ | $v_{12}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $x^{(0)}$ | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| $x^{(1)}$ | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| $x^{(2)}$ | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| $x^{(3)}$ | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| $x^{(4)}$ | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| **Sum:** | **1** | **1** | **2** | **1** | **1** | **2** | **1** | **1** | **2** | **1** | **1** | **2** |

**Figure 10** An orbit of ECA rule 1 with update order $\pi = 12\cdots n$. Note that the period of the sum vector $(1, 1, 2, 1, 1, 2, 1, 1, 2, 1, 1, 2)$ is 3, as guaranteed by Proposition 10.

The sum vector of the orbit table in Figure 8 has period 1, as each column contains exactly two live entries. See Figure 10 for an orbit table with a sum vector of period 3. The proof of Proposition 10 follows from translating into ticker tape notation, the fact that the standard slither must have an odd number of $D$s. This last fact is due to the relation in Eq. (3) with the observation that the shadow function always skips exactly two rows, but

loses one when it wraps around the end of the table. For example, in Figure 10, where $s^6 = c^2$, we can travel from position $(0, 1)$ to $(3, 10)$ via the standard slither $(D2)^3$ or the standard co-slither $SL$. Both $S$ and $L$ skip two rows, but we lose one with the $S$ because it wraps back to $(1, 12)$.

It is worth noting that Proposition 10 does not necessarily hold for other update orders. For example, if $n = 4$ with update order $\pi = 1243$, there is a size-2 orbit $\{0100, 0001\}$ whose sum vector $(0, 1, 0, 1)$ has period 2. More generally, the methods and theory developed here do not carry over seamlessly to toggling independent sets under other update orders, but we are working on adapting them. It helps that for many problems, we only need to consider the $\lceil \frac{n}{2} \rceil$ update orders described in Eq. (2).

By knowing that the standard slither must contain an odd number of $D$s, we can construct and characterize all possible scrolls and orbit tables. Without loss of generality, we can assume that the "first entry," i.e., cell 1 at $t = 0$, is live.

▶ **Theorem 11.** *For a fixed $n$, we can construct all scrolls having $X_{0,1} = 1$ through the following procedure:*

1. *Take a solution to the equation*

$$2a + 3b + 4c = n + 1 \quad \text{with} \quad a, b, c \in \mathbb{Z}_{\geq 0} \text{ and } b + c > 0. \tag{4}$$

2. *Choose any sequence of $2(b + c) - 1$ instances of $D$ and $a$ instances of 2. This gives the standard slither of each snake.*
3. *Choose any sequence of $b$ instances of $S$, and $c$ instances of $L$. This gives the standard co-slither of each co-snake.*

*Furthermore, each solution descends to a unique orbit table under the quotient map $p$ (though some of these orbit tables may arise from ticker tapes that differ only by translation).*

Being able to characterize all possible scrolls for a fixed $n$ by constructing their standard slithers and co-slithers also tells us the possible snake and ouroboros groups. In other words, it gives essential information about the feasible topological structures on the set of live entries. It should be possible to say more for special cases, such as when $n$ is prime, even, odd, etc. This would relate fundamental topological and dynamical properties of ECA rule 1.

## 3 Ongoing and future work

This paper arose out of our work on toggling independent sets from combinatorics, the connections we saw to cellular automata, and how both fields complement each other. Our original goal was to prove our data-driven conjecture that the period of the sum vector is always odd under the ACA map $F_{12\cdots n}$. We expected this to lead us to explore homomesies and how the dynamics behaves under toric equivalent update orders, because these questions arose in earlier work of toggling independent sets over the path graph [10]. However, once we encountered the algebraic and topological structures that arose from the commuting successor and shadow functions, our research went in a different direction.

Many of the general ideas in this paper, such as studying tables of 0s and 1s in a CA, are obviously not new, and have been around for decades. Nevertheless, there are many novel approaches within our techniques that we think should open up new avenues for research within the fields of both combinatorics and cellular automata. We would still like to return to studying homomesy, resonance, and toric equivalence in ECA rule 1. We would also like to investigate this over other families of graphs, such as the "distance-2 cycle graph". In the literature, some of this is done under the name of "SDSs with logical NOR functions" [15].

However, to our knowledge, these have not been studied using the algebraic framework in this paper. More generally, studying homomesy and resonance in other asynchronous cellular automata beyond rule 1, and relating them to the algebraic features of toggle groups and the combinatorial features of the Coxeter elements, is mostly unexplored. This is an example of theory from dynamical algebraic combinatorics contributing to the field of cellular automata.

For another example, the idea behind the shadow and successor functions can be generalized to other ECA rules. Namely, if $X_{i,j} = 1$, then we will call positions $(i', j')$ and $(i'', j'')$ in a scroll a **complementary pair** of $(i, j)$ if $X_{i',j'} + X_{i'',j''} = 1$. Given a live position $(i, j)$, every complementary pair defines a function that outputs the position of the unique live entry. By Proposition 3, the successor and shadow functions are **commuting complementary pairs**. This defines the abelian "snake group," and it guarantees that all (co-)snakes have a common (co-)slither, because they arise as cosets. The natural simply transitive actions of these groups define Cayley diagram structures on the live entries of the scrolls and orbit tables, and a topological covering map relating the two. We have a paper in preparation, which also includes the proofs of many results stated here, that uses the theory of covering spaces and deck transformations to deduce properties about the periodicity within the scrolls and ticker tapes. It would be interesting to find examples of complementary pairs arising from other ECA rules, either under a fixed sweep $F_\pi$, or synchronous update. If two such pairs commute, then this opens to door to using algebraic and topological tools to analyze those dynamical systems. Even if they do not commute, it is still likely that much of the framework still carries over, but with a nonabelian group.

The complementary pairs that define the successor and shadow functions are characterized by $X_{i,j+2} + X_{i+1,j+1} = 1$ and $X_{i+2,j-2} + X_{i+2,j-1} = 1$. We can try to directly define other commuting complementary pairs by modifying these indices, and investigate which ones give rise to infinite binary scrolls of width $n$, that a so-called "generalized snake group" acts on simply transitively. When this happens, we can ask whether or not such a scroll arises from a togglable CA, SDS, or automata network (i.e., possibly over a different graph), how to characterize those that do, and whether all of the orbits can be specified in a number-theoretic manner, such as in Theorem 11.

Of course, these ideas can be extended beyond just pairs – e.g., to a set of 3 entries that must contain exactly 1 (or 2) live entries. Scrolls of ECA rule 1 with $\pi = 12 \cdots n$ have two commuting complementary pairs, but rules that are less sparse might have scrolls that are characterized by three or more. It seems like if this were to happen, the resulting three-generator abelian group would not act freely on the live entries, because of the two-dimensional structure of a scroll. However, this is all speculative, and deserves investigation.

The previous examples are all instances of ideas from dynamical algebraic combinatorics posing new questions in the field of cellular automata. On the other hand, ideas from CAs can lead to new problems in combinatorics. For example, it is easy to define what it means to toggle other graph-theoretic objects, such as vertex covers or dominating sets. Over the cycle graph $\mathcal{C}_n$, each of these can also be viewed as a CA, albeit a distance-2 one. Though it is more complicated, the idea of complementary pairs should be adaptable for certain toggling problems that do not involve CAs, such as order ideal or antichain toggling. This amounts to finding invariants of an arbitrary live entry at time $t$, in terms of a pair of entries at, e.g., time $t'$ and time $t''$. Each such pair will define a bijection on the live entries and thereby generate an equivalence relation. This is just one example of how it might be feasible to adapt tools developed from analyzing tables of 0s and 1s arising from ECA rule 1 to new techniques applicable to generalized toggling problems in combinatorics.

## References


1 D. Armstrong, C. Stump, and H. Thomas. A uniform bijection between nonnesting and noncrossing partitions. *Trans. Amer. Math. Soc.*, 365(8):4121–4151, 2013.
2 A. Björner and F. Brenti. *Combinatorics of Coxeter groups*. Springer Verlag, New York, 2005.
3 P. Cameron and D. Fon-Der-Flaass. Orbits of antichains revisited. *European J. Combin.*, 16(6):545–554, 1995.
4 L. David, C. Defant, M. Joseph, M. Macauley, and A. McDonough. Toggling independent sets in cycle graphs, 2021. In preparation.
5 C. Defant. Flexible toggles and symmetric invertible asynchronous elementary cellular automata. *Discrete Math.*, 341(9):2367–2379, 2018.
6 K. Dilks, O. Pechenik, and J. Striker. Resonance in orbits of plane partitions and increasing tableaux. *J. Comb. Theory Ser. A*, 148:244–274, 2017.
7 D. Einstein, M. Farber, E. Gunawan, M. Joseph, M. Macauley, J. Propp, and S. Rubinstein-Salzedo. Noncrossing partitions, toggles, and homomesies. *Electron. J. Combin.*, 23(3), 2016.
8 H. Eriksson and K. Eriksson. Conjugacy of Coxeter elements. *Electron. J. Combin.*, 16(2):#R4, 2009.
9 M. Joseph. Antichain toggling and rowmotion. *Electron. J. Combin.*, 26(1), 2019.
10 M. Joseph and T. Roby. Toggling independent sets of a path graph. *Electron J. Combin.*, 25(1):1–18, 2018.
11 G. Kreweras. Sur les partitions non croisées d'un cycle. *Discrete Math.*, 1(4):333–350, 1972.
12 M. Macauley, J. McCammond, and H. Mortveit. Dynamics groups of asynchronous cellular automata. *J. Algebraic Combin.*, 33(1):11–35, 2011.
13 M. Macauley, J. McCammond, and H.S. Mortveit. Order independence in asynchronous cellular automata. *J. Cell. Autom.*, 3(1):37–56, 2008.
14 M. Macauley and H.S. Mortveit. Cycle equivalence of graph dynamical systems. *Nonlinearity*, 22:421–436, 2009.
15 H.S. Mortveit and C.M. Reidys. Discrete, sequential dynamical systems. *Discrete Math.*, 226(1-3):281–295, 2001.
16 H.S. Mortveit and C.M. Reidys. *An Introduction to Sequential Dynamical Systems*. Universitext. Springer Verlag, 2007.
17 Y. Numata and Y. Yamanouchi. On the action of the toggle group of the Dynkin diagram of type A. *arXiv:2103.16217*, 2021.
18 O. Pretzel. On reorienting graphs by pushing down maximal vertices. *Order*, 3(2):135–153, 1986.
19 J. Propp and T. Roby. Homomesy in products of two chains. *Electron. J. Combin.*, 22(3), 2015.
20 T. Roby. Dynamical algebraic combinatorics and the homomesy phenomenon. In *Recent Trends in Combinatorics*, pages 619–652. Springer, 2016.
21 V. Salo. Universal gates with wires in a row. *arXiv:1809.08050*, 2018.
22 B. Schönfisch and A. de Roos. Synchronous and asynchronous updating in cellular automata. *BioSystems*, 51(3):123–143, 1999.
23 M. Schützenberger. Promotion des morphismes d'ensembles ordonnés. *Discrete Math.*, 2(1):73–94, 1972.
24 J. Striker. Dynamical algebraic combinatorics: promotion, rowmotion, and resonance. *Notices Amer. Math. Soc.*, 64(6), 2017.
25 J. Striker. Rowmotion and generalized toggle groups. *Discrete Math. Theor. Comput. Sci.*, 20, 2018.
26 J. Striker and N. Williams. Promotion and rowmotion. *European J. Combin.*, 33:1919–1942, 2012.
27 M. Vielhaber. Computing by temporal order: asynchronous cellular automata. In *AUTOMATA*, volume 90, pages 166–176. Electron. Proc. Theor. Comput. Sci., 2012.

# Rice's Theorem for Generic Limit Sets of Cellular Automata

**Martin Delacourt** ✉
Université d'Orléans, LIFO EA4022, FR-45067 Orléans, France

## Abstract

The generic limit set of a cellular automaton is a topologically defined set of configurations that intends to capture the asymptotic behaviours while avoiding atypical ones. It was defined by Milnor then studied by Djenaoui and Guillon first, and by Törmä later. They gave properties of this set related to the dynamics of the cellular automaton, and the maximal complexity of its language. In this paper, we prove that every non trivial property of these generic limit sets of cellular automata is undecidable.

**2012 ACM Subject Classification** Theory of computation → Models of computation

**Keywords and phrases** cellular automata, dynamical systems, generic-limit sets, Rice's theorem, subshifts

**Digital Object Identifier** 10.4230/OASIcs.AUTOMATA.2021.6

## 1 Introduction

Cellular automata (CA) are discrete dynamical systems defined by a local rule, introduced in the 40s by John von Neumann [13]. Given a finite alphabet $\mathcal{A}$, the global rule on $\mathcal{A}^{\mathbb{Z}}$ is given by the synchronous application of the local one at every coordinate. They can be seen as models of computation, dynamical systems or many phenomena from different fields, providing links between all of these [5, 9].

The asymptotic behaviour of CA has been studied a lot, mainly using the definition of limit set: the set of points that can be observed arbitrarily far in time. In particular concerning the complexity of this set: it can be non-recursive, the nilpotency problem is undecidable and there is Rice's theorem on properties of the limit set of CA [6, 7, 8]. Rice's theorem states that every nontrivial property of the limit set of CA is undecidable. Other definitions were introduced in order to restrain to typical asymptotic behaviour. Milnor proposed the definition of likely limit set and generic limit set in [11] in the more general context of dynamical systems. While the likely limit set is defined in the measure-theoretical world, the generic limit set is a topological variant. Djenaoui and Guillon proved in [4] that both are equal for full-support $\sigma$-ergodic measures in the case of CA.

The generic limit set is the smallest closed subset of the fullshift $\Sigma^{\mathbb{Z}}$ containing all limit points of all configurations taken in a comeager subset of $\Sigma^{\mathbb{Z}}$. Djenaoui and Guillon studied the generic limit set in [4], proving results on the structure of generic limit sets related to the directional dynamics of CA. They also provide a combinatorial characterization of the language of the generic limit sets and examples of CA with different limit, generic limit and $\mu$-limit sets. The latter was introduced in [10] by Kůrka and Maass as another measure-theoretical version of limit set.

The $\mu$-limit set is determined by its language which is the set of words that do not disappear in time, relatively to the measure $\mu$. Amongst the results on the $\mu$-limit set, it was proved in [1] that the complexity of the language is at the level 3 of the arithmetical hierarchy ($\Sigma_3^0$), with a complete example, it was also proved that the nilpotency problem is $\Pi_3^0$-complete. Rice's theorem also holds stating that each nontrivial property has at least $\Pi_3^0$ complexity. A slightly different approach led Hellouin and Sablik to similar results on the limit probability measure in [2].

In [12], Törmä proved computational complexity results on the generic limit sets, in particular an example of a CA with a $\Sigma^0_3$-complete generic limit set, and constraints on the complexity when the dynamics of the CA is too simple on the generic limit set.

In this paper, we prove Rice's theorem on generic limit sets combining ideas from [8] and [1].

## 2 Definitions

In this paper, we consider the countable set $\mathcal{Q} = \{q_0, q_1, q_2, \dots\}$. Every finite alphabet will be a finite subset of $\mathcal{Q}$. Given a finite alphabet $\Sigma \subseteq \mathcal{Q}$ and a radius $r \in \mathbb{N}$, a local rule is a map $\delta : \Sigma^{2r+1} \to \Sigma$ and a *cellular automaton* $\mathcal{F} : \Sigma^{\mathbb{Z}} \to \Sigma^{\mathbb{Z}}$ is the global function associated with some local rule $\delta$: for every $c \in \Sigma^{\mathbb{Z}}$ and every $i \in \mathbb{Z}$, $\mathcal{F}(c)_i = \delta(c_{i-r}, c_{i-r+1}, \dots, c_{i+r})$. We call *configurations* the elements of $\Sigma^{\mathbb{Z}}$. The orbit of an initial configuration $c$ under $\mathcal{F}$ is called a *space-time diagram*. Time goes upward in the illustrations of this paper.

Define the Cantor topology on $\Sigma^{\mathbb{Z}}$ using the distance $d(c, c') = \frac{1}{2^i}$ where $i = \min\{j \in \mathbb{N}, c_j \neq c'_j \text{ or } c_{-j} \neq c'_{-j}\}$. For any word $w \in \Sigma^*$, denote $|w|$ the length of $w$ and $[w]_i = \{c \in \Sigma^{\mathbb{Z}} : \forall k < |w|, c_{i+k} = w_k\}$ the associated *cylinder set*, which is a clopen set.

Denote $\sigma$ the shift on $\Sigma^{\mathbb{Z}}$, which is the CA such that $\forall c \in \Sigma^{\mathbb{Z}}, \forall i \in \mathbb{Z}, \sigma(c)_i = c_{i+1}$. A *subshift* is a closed $\sigma$-invariant subset of $\Sigma^{\mathbb{Z}}$. A subshift can be equivalently defined by the set of forbidden words, in this case a subshift is the set of configurations that do not belong to any $[w]_i$ where $w$ is forbidden.

In this paper, a Turing machine works on a semi-infinite (to the right) tape, with a finite alphabet $\mathcal{A}$ containing a blank symbol $\perp$. It has one initial state $q_0$ and one final state $q_f$. At each step of the computation, the head of the machine reads the symbol at the position on the tape to which it points, and decides the new symbol that is written on the tape, the new state it enters, and its move (one cell at most). It can be simulated by a CA using states that can contain the head of the machine and the tape alphabet. We will here only simulate machines in a finite space in which there is only one head.

### 2.1 Limit sets of cellular automata

Different definitions of the asymptotic behavior of a CA have been given. The most classical one is the *limit set* $\Omega_{\mathcal{F}} = \bigcap_{t \in \mathbb{N}} \mathcal{F}^t(\Sigma^{\mathbb{Z}})$ of a CA $\mathcal{F}$, that is the set of configurations that can be seen arbitrarily late in time. For any subset $X \subseteq \Sigma^{\mathbb{Z}}$, define $\omega(X)$ as the set of limit points of orbits of configurations in $X$: $c \in \omega(X) \Leftrightarrow \exists c' \in X, \liminf_{t \to \infty} d(\mathcal{F}^t(c'), c) = 0$. The set $\omega(\Sigma^{\mathbb{Z}})$ is called the *asymptotic set* of $\mathcal{F}$.

A subset $X \subseteq \Sigma^{\mathbb{Z}}$ is said to be *comeager* if it contains a countable intersection of dense open sets. It implies in particular that $X$ is dense (Baire property).

For $X \subseteq \Sigma^{\mathbb{Z}}$, define the *realm of attraction* $\mathcal{D}(X) = \{c \in \Sigma^{\mathbb{Z}} : \omega(c) \subseteq X\}$. The *generic limit set* $\tilde{\omega}(\mathcal{F})$ of $\mathcal{F}$ is then defined as the intersection of all closed subsets of $\Sigma^{\mathbb{Z}}$ whose realms of attraction are comeager.

The following two examples show differences between all these sets, they were already presented in [4].

▶ **Example 1** (The Min CA). Consider the CA $\mathcal{F}$ of radius 1 on alphabet $\{0, 1\}$ whose local rule is $(x, y, z) \mapsto \min(x, y, z)$. The state 0 is spreading, that is, every cell that sees this state will enter it too. A space-time diagram of the MIN CA is represented in Figure 1.

We have:

- $\Omega_{\mathcal{F}} = \{c \in \{0, 1\}^{\mathbb{Z}} : \forall i \in \mathbb{Z}, k \in \mathbb{N}^*, c \notin [10^k 1]_i]\}$;
- $\tilde{\omega}(\mathcal{F}) = \{0^{\mathbb{Z}}\}$ and it is equal to the $\mu$-limit set for a large set of measures containing every non degenerate Markov measure.

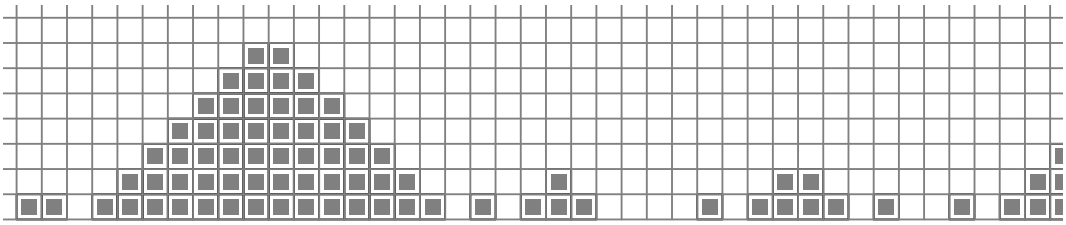


**Figure 1** Some part of a space-time diagram of the Min CA, 0 is represented by the white state and 1 by the black state.

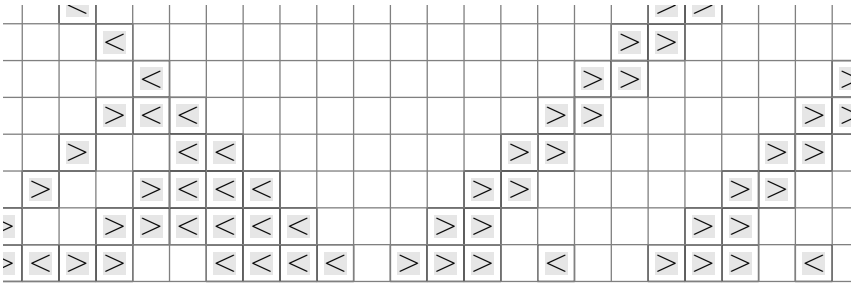


**Figure 2** The $<$ and $>$ states of the Gliders CA are particles going in different directions and annihilating each other when they cross.

▶ **Example 2** (Gliders). Consider the CA $\mathcal{F}$ of radius 1 on alphabet $\{0, >, <\}$. The states $<$ and $>$ are respectively speed $-1$ and 1 signals over a background of 0s. When a $<$ and a $>$ cross, they both disappear. A space-time diagram of this CA is represented in Figure 2. For a complete description of the rule, see for example [10, Example 3].

We have:

- $\Omega_{\mathcal{F}} = \{c \in \{0, <, >\}^{\mathbb{Z}} : \forall i \in \mathbb{Z}, k \in \mathbb{N}, c \notin [< 0^k >]_i]\}$;
- $\tilde{\omega}(\mathcal{F}) = \Omega_{\mathcal{F}}$;
- the $\mu$-limit set depends here of $\mu$. With $\mu$ the uniform Bernoulli measure, it is $\{0^{\mathbb{Z}}\}$. If $\mu$ is Bernoulli with a bigger probability for $<$ than for $>$, then the $\mu$-limit set is $\{\{0, <\}^{\mathbb{Z}}\}$.

## 2.2 Preliminary properties of generic limit sets of CA

Many properties of generic limit sets were proved either in [11] or in [4] for the particular case of CA.

▶ **Proposition 3** (Prop 4.2 of [4]). *Given a CA $\mathcal{F}$, the realm of attraction of $\tilde{\omega}(\mathcal{F})$ is comeager.*

▶ **Proposition 4** (Prop 4.4 of [4]). *Given a CA $\mathcal{F}$, $\tilde{\omega}(\mathcal{F})$ is a subshift.*

Note that the limit set of a CA is also a subshift whereas the asymptotic limit set may not be.

▶ **Proposition 5** (Cor 4.7 of [4]). *Given a CA $\mathcal{F}$ on alphabet $\Sigma$, $\tilde{\omega}(\mathcal{F}) = \Sigma^{\mathbb{Z}} \Leftrightarrow \mathcal{F}$ is surjective.*

The last result of this section comes from Remark 4.3 of [4] and is reformulated as Lemma 2 of [12]:

▶ **Lemma 6.** *Let $\mathcal{F}$ be a CA on $\Sigma^{\mathbb{Z}}$. A word $s \in \Sigma^*$ occurs in $\tilde{\omega}(\mathcal{F})$ if and only if there exists a word $v \in \Sigma^*$ and $i \in \mathbb{Z}$ such that for all $u, w \in \Sigma^*$, there exist infinitely many $t \in \mathbb{N}$ with $\mathcal{F}^t([uvw]_{i-|u|}) \cap [s] \neq \emptyset$.*

The word $v$ is said to *enable* $s$.

# 3 General structure of the construction

The proof of the main result of this paper relies on a construction already presented in [3, 1, 2]. The present section contains the description of this tool. The idea is to erase most of the content of the initial configuration and start a protected (hence controled) and synchronized evolution. Of course, to ensure that this property holds for any configuration, one needs strong constraints on the dynamics of the CA. Here, we also want to allow a wide variety of dynamics, hence this property shall hold for almost every initial configuration. In the above-cited articles, it was true for $\mu$-almost every configuration, and here we will use a topological variant.

A brief description of this CA $\mathcal{F}$ follows. Its radius should be at least 2.

## 3.1 Overview

Some particular state $\boxed{*} \in \Sigma$ can only appear in the initial configuration: there is no rule that produces it. The states $\boxed{*}$ will trigger the desired evolution. In order to avoid having to deal with anything unwanted on the initial configuration (like words produced by the evolution of the CA placed in a wrong context), we add a mechanism that cleans the configuration from anything that is not produced by $\boxed{*}$. This is achieved through the propagation of large signals that have the information of the time passed since a $\boxed{*}$ state produced it, that is their age. Then, when two such signals going in opposite directions meet, they compare their ages and only the younger survives.

With this trick, any configuration that contains infinitely many $\boxed{*}$ on both sides will ultimately be covered by protected areas. The $\boxed{*}$ states also transform into $\boxed{\#} \in \Sigma$ states, and we consider the words in the space-time diagram that are delimited by $\boxed{\#}$ states produced by $\boxed{*}$ states, we call them segments. The dynamics of the CA inside a segment only depends on its size. In particular, the simulation of the computation of a given Turing machine can be started on each $\boxed{\#}$ state when it appears.

A close construction with a more precise and complete description can be found in [1, Section 3.1].

## 3.2 Initialization and counters

The state $\boxed{*}$ can only appear in the initial configuration: it is not produced by any rule and it disappears immediately. Consider a cell at coordinate $i$ that contains a $\boxed{*}$ state in the initial configuration. On each side of the $\boxed{*}$ state, two signals are sent at speed $s_f$ and $s_b$ to the right and symmetrically to the left. The fastest one (speed $s_f$) erases everything it encounters except for its symmetrical counterpart. Each couple of signals is seen as one counter whose value is encoded by the distance $\lfloor k(s_f - s_b) \rfloor$ after $k$ steps of the CA. The key point is that, at any time, the value of a counter is minimal exactly for counters generated by a $\boxed{*}$ state.

When two counters meet, they compare their values without being affected until the comparison is done. The comparison process is done via signals bouncing on the borders of the counters. The speed of these inner signals is greater than the speeds ($s_f$ and $s_b$) of the border signals. As the value is encoded by the distance between border signals, it is a geometric comparison illustrated in Figure 3. If one counter is younger than the other one, the older one is deleted (the right one in Figure 3). If they are equal, both counters, that is the 4 signals, are deleted.

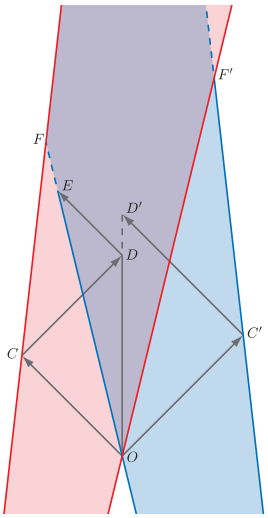



**Figure 3** When counters meet in $O$, signals move at speed 1 towards the borders of the counters that they reach at points $C$ and $C'$. They bounce back until they cross the sign left at point $O$. The one that arrives first has crossed the most narrow (hence youngest) one. It bounces once again to erase the opposite counter whose border is reached at point $E$.

▷ Claim 7. For any configuration $c$ where $\boxed{*}$ occurs, and any coordinate $i \in \mathbb{Z}$, denote $d_i = \min\{|i-j| : c_j = \boxed{*}\}$. Then for any $t > s_b d_i$ (where $s_b$ is the speed of the inner border of the counter), $\mathcal{F}^t(c)_i$ does not contain a counter state.

Proof. Each sequence of consecutive $\boxed{*}$ states creates a left counter at its left extremity and a right counter at its right extremity. They all share a common age which is the minimal one, hence they cannot be crossed by another counter. Thus, at most one of the youngest counters can cross cell $i$. And due to the speed of the inner border of the counters, this is done after $s_b d_i$ steps. ◁

Last rule of this construction: every $\boxed{*}$ state that is not surrounded by other $\boxed{*}$ states on both sides is replaced by a $\boxed{\#}$ state after it gave birth to the counters. Figure 4 shows how a typical initial configuration evolves.

For any time $t \in \mathbb{N}$ and any configuration $c$, we call *segment* a set of consecutive cells from coordinate $i$ to $j$ in $\mathcal{F}^t(c)$ with $i, j \in \mathbb{Z}$ such that:

- $\mathcal{F}^t(c)_i = \boxed{\#} = \mathcal{F}^t(c)_j$
- for every $i < k < j$, $\mathcal{F}^t(c)_k \neq \boxed{\#}$
- $c_i = \boxed{*}$ and $c_j = \boxed{*}$.

Note that if the radius of the CA can be arbitrarily large, any choice of speeds $s_f > s_b$ can be made.

▷ Claim 8. For any $s \in \mathbb{Q}$, there exists a CA implementing such a construction with speed $s_b > s$ (and hence $s_f$).

Proof. A big enough radius allows fast enough signals to perform the comparison of counters in due time. ◁

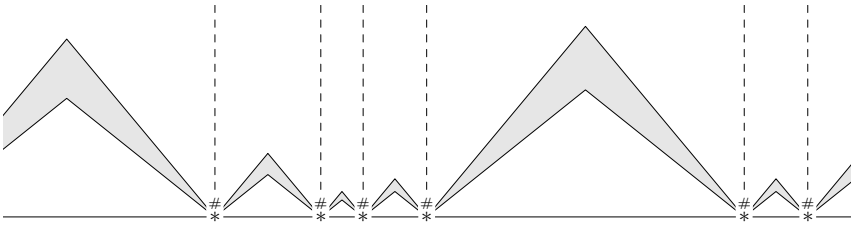


**Figure 4** Starting from a configuration containing infinitely many $\boxed{*}$ states on the left and on the right, the $\boxed{*}$ states generate counters (filled in grey) on both sides that erase everything but another counter going in the opposite direction. These counters eventually meet their opposite and disappear after comparing their ages, hence remain an immaculate configuration with $\boxed{\#}$ states in some positions.

## 4 Rice's theorem

Following the steps of the historical proof of Rice and concerning CA, the theorems on limit sets in [8] and $\mu$-limit sets in [3], we first define properties of generic limit sets of CA, then prove that every non trivial such property is undecidable.

The CA used in [3] to prove Rice's theorem for $\mu$-limit sets also has the general structure presented in the previous section. The difference lies in what is done inside segments. In the case of $\mu$-limit sets (regardless of the choice of $\mu$), it is possible to dedicate a small *technical* space inside segments to any activity that shouldn't appear in the $\mu$-limit set, as long as this space tends to disappear in density. This is achieved through larger and larger segments. Nothing prevents the states of this technical space to appear in the generic limit set.

### 4.1 Properties of generic limit sets of CA

A property of the generic limit set of CA is a set of subshifts and we say that a generic limit set have this property if it belongs to this set. This way, it depends only on the generic limit set: if two CA have the same generic limit set, this common generic limit set either has or not the property. As mentionned earlier, we consider the countable set $\mathcal{Q} = \{q_0, q_1, \dots\}$, and every alphabet is a finite subset of $\mathcal{Q} = \{q_0, q_1, \dots\}$.

▶ **Definition 9.** *A property $\mathcal{P}$ of generic limit sets of cellular automata is a subset of the powerset $\mathscr{P}(\mathcal{Q}^{\mathbb{Z}})$. A generic limit set of some cellular automaton is said to have property $\mathcal{P}$ if it is in $\mathcal{P}$.*

Note that many sets that are not subshifts can belong to a property $\mathcal{P}$, as every generic limit set is a subshift, they do not matter. In particular, every property that does not contain a subshift is equivalent to the empty property that no generic limit set has. A property is said to be *trivial* when either it contains all generic limit sets or none. The most natural example of a non trivial property is the *generic nilpotency*, which is given by the family $\{\{q_i^{\mathbb{Z}}\}, i \in \mathbb{N}\}$.

This definition prevents confusions between properties of generic limit sets and properties concerning generic limit sets. For example the property containing every fullshift on finite alphabets is not surjectivity, since the generic limit set of a CA on alphabet $\Sigma$ could be a fullshift on a strictly smaller alphabet. Hence surjectivity is not a property of generic limit sets even if being surjective is equivalent to having a full generic limit set. .

## 4.2 The theorem

▶ **Theorem 10.** *Every non trivial property of the generic limit sets of CA is undecidable.*

This section is dedicated to the proof of Rice's theorem. It is a many-one (actually one-one) reduction from the Halting problem on empty input for Turing machines. Take a non trivial property $\mathcal{P}$ of generic limit sets of CA. Assume for example that $\mathcal{P} \cap \{\{q_k^{\mathbb{Z}}\}, k \in \mathbb{N}\}$ is infinite (the other case leads to a symmetric proof). As $\mathcal{P}$ is non trivial, it is possible to choose $q_n \in \mathcal{Q}$ and a CA $\mathcal{F}_1$ such that $\tilde{\omega}(\mathcal{F}_1) \notin \mathcal{P}$ and $q_n \notin \Sigma_1$ where $\Sigma_1$ is the alphabet of $\mathcal{F}_1$. Denote now $\mathcal{F}_0$ the CA on alphabet $\{q_n\}$ whose local rule always produces $\{q_n\}$. Hence $\tilde{\omega}(\mathcal{F}_0) = \{q_n^{\mathbb{Z}}\} \in \mathcal{P}$.

For any Turing machine $M$, we produce a CA $\mathcal{F}_M$ such that:

- if $M$ eventually halts on empty input, the generic limit set of $\mathcal{F}_M$ is $\{q_n^{\mathbb{Z}}\}$;
- if $M$ never halts on empty input, then the generic limit set of $\mathcal{F}_M$ is $\tilde{\omega}(\mathcal{F}_1)$.

### 4.2.1 Construction of $\mathcal{F}_M$

The CA $\mathcal{F}_M$ contains two layers, one for each of the main tasks. Denote $\pi_1$ and $\pi_2$ the projections on the first and second layer. The first layer uses alphabet $\Sigma_0$ and it implements the construction described in Section 3. Denote _ the blank state of $\Sigma_0$. The second layer simulates the CA $\mathcal{F}_1$. In some cases, the first layer can be erased, we also add a state $q_n$, hence the alphabet of $\mathcal{F}_M$ is $\Sigma = (\Sigma_0 \times \Sigma_1) \cup \{q_n\} \cup \Sigma_1$.

The set $\Sigma_0 \times \Sigma_1$ can be mapped to a subset of $\mathcal{Q} \setminus (\{q_n\} \cup \Sigma_1)$ to ensure that $\Sigma \subset \mathcal{Q}$. For the clarity of the presentation, we will denote the elements of $\Sigma_0 \times \Sigma_1$ as couples.

The idea is to let $\mathcal{F}_1$ compute on the second layer (or by itself if the first layer has been erased), while computation on the first layer will either lead to erase this layer or generate a $q_n$ state that will be spreading (erasing everything but counters) over the whole configuration.

On the first layer, once a $\boxed{\#}$ state appears (from a $\boxed{*}$ state), a simulation of $M$ is started on its right. In the general case, another $\boxed{\#}$ state exists further on the right, in which case this simulation takes place in a segment. We will show later that the other case is irrelevant when considering the generic limit set. The simulation evolves freely except if it is blocked by the inner border of a counter, if this happens the simulated Turing head waits until it has enough space to make one more step. A binary counter is started in parallel to the right of the $\boxed{\#}$ state.

The simulation inside a segment should always be finite, it can be interrupted for one of the following reasons.

- The simulation of $M$ halts (because $M$ reaches a final state). Then the state $q_n$ is written, erasing both layers of $\mathcal{F}_M$. This state spreads to both of its neighbors erasing everything, even the $\boxed{\#}$ states, except for the inner and outer borders of the counters of the construction of Section 3.
- It reaches a $\boxed{\#}$ on its right. That is there is not enough space inside the segment and the simulation is aborted. The first layer content of the segment will be erased as explained later.
- The counter reaches another $\boxed{\#}$ state. The time allowed for the simulation is over and the simulation is aborted. This third case is necessary to avoid problems due to a loop of the Turing machine in a finite space.

The states used for the simulation should not appear in the generic limit set, hence they have to be erased once the simulation halts or is aborted. In the first case, the state $q_n$ is written in every cell. In the second case, the first layer only is erased. For the same reason, the $\boxed{\#}$ state has to be erased when the simulation is over in both the segments it delimits.

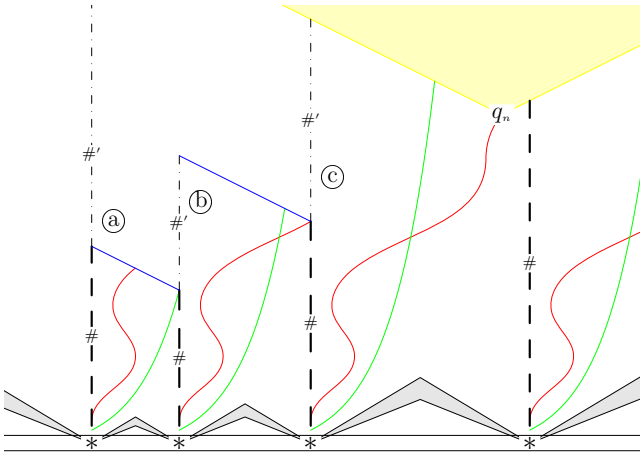



**Figure 5** Starting from the cells in state $\boxed{*}$ in the initial configuration, the counters (grey areas) protect everything above them. Segments are delimited by $\boxed{\#}$ states and in each of them a simulation of the computation of a Turing machine takes place (the red curve gives the position of the head). The green curve represents the extension of the binary counter used to limit the time of the simulation. In segment ⓐ, the counter reaches the limit and an abortion signal is sent (blue). In segment ⓑ, the head reaches the right of counter and the simulation is stopped with an abortion signal sent to the left. In segment ⓒ, the Turing machine halts and the spreading state $q_n$ is written.

If the simulation is aborted (due to lack of space or end of the allowed time in the segment), an *abortion signal* is sent in both directions that erases everything of the first layer (except outer or inner border of counters) until it reaches a $\boxed{\#}$ state. A $\boxed{\#}$ state that receives such an abortion signal transforms into a $\boxed{\#'}$ state. If a $\boxed{\#'}$ state receives an abortion signal, it disappears. The point is to ensure that the abortion signals do not travel too far: if the first abortion signal deletes the $\boxed{\#}$ state on the side of the segment, then the one arriving from the other side will cross. This could lead to the presence of abortion signals in the generic limit set.

Figure 5 is a schematic view of the evolution of CA $\mathcal{F}_M$ on an ordinary initial configuration.

▷ Claim 11. There exists an increasing function $f : \mathbb{N} \to \mathbb{N}$ such that the computation of $M$ simulated in a segment of length $n$ either halts or is aborted before time $f(n)$.

Proof. In a segment of length $n$, due to the binary counter, if the simulation of $M$ has not reached a final state after $2^n$ steps, the computation is aborted. ◁

### 4.2.2 Ensuring a sound computation on the second layer

The proof relies on the fact that, with most initial configurations:

- if $M$ halts, there will exist a large enough segment in which the computation has enough space and time to reach its end, thus producing state $q_n$ that erases everything.
- if $M$ does not halt, the computation will be eventually aborted in every segment and only the second layer will remain with a computation of $\mathcal{F}_1$.

In order to ensure the second point, we need to deal with the case of $q_n$ states existing before the counters of Section 3 clean the configuration on the first layer. It can for example happen due to $q_n$ states on the initial configuration. In this case, the content of the second layer is lost. As it is impossible to control what happens outside the area protected by counters, the counters will not only stop the spreading of $q_n$ but also write a possible configuration for $\mathcal{F}_1$, thus deleting all data that does not descend from the cells containing $\boxed{*}$ in the first layer of the initial configuration.

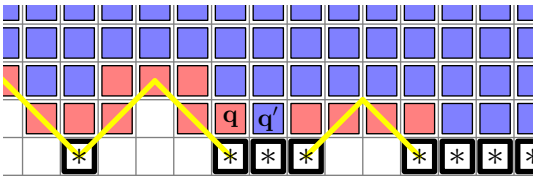



**Figure 6** Partial representation of a space-time diagram of $\mathcal{F}_M$. The red cells are where the counters rewrite the second layer assuming that what does not come from a $\boxed{*}$ state is $x_0$. The blue cells are where the computation of $\mathcal{F}_1$ happens normally on the second layer. The yellow lines are the outer borders of counters, we assume here they have speed 1 for the illustration. Denote $\delta_1$ the local rule of $\mathcal{F}_1$. Then $q' = \delta_1(x, y, z)$ which are its state ($y$) and the ones of its neighbors ($x$ and $z$) at time 0. And $q = \delta_1(x_0, x, y)$.

Let us assume for simplicity that the radius of $\mathcal{F}_1$ is 1. For the rest of the proof of the theorem, denote $x_0$ some state of $\Sigma_1$. The space-time diagram of $\mathcal{F}_1$ with initial configuration $x_0^{\mathbb{Z}}$ is ultimately periodic, contains only uniform configurations and is entirely described by a finite sequence of distinct states $(x_0, x_1, \dots, x_p, \dots, x_{p+T}, x_p)$. The counters will write the second layer of the configuration as if every information coming from outside the protected area (between counters) was obtained from the uniform initial configuration $x_0^{\mathbb{Z}}$:

- $x_t$ at step $t \leq p$;
- $x_{p+(t-p) \mod T}$ at step $t \geq p$.

As a finite amount of information is needed, the local rule of the CA $\mathcal{F}_M$ can be designed to do so. This is illustrated in Figure 6. As said in Claim 8, it is possible to use that construction with outer borders of counters moving at speed 1.

If the first layer contains $\boxed{*}$, the state on the second layer is not rewritten and is used for the simulation of $\mathcal{F}_1$.

To any initial configuration $x \in \Sigma^{\mathbb{Z}}$, corresponds a configuration in $\Sigma_1^{\mathbb{Z}}$ where all the deleted data is replaced by $x_0$. Denote $\phi : \Sigma \to \Sigma_1$ such that:

- $\phi(\boxed{*}, x) = x$;
- $\phi(s, x) = x_0$ when $s \neq \boxed{*}$;
- $\phi(x) = x_0$ when $x \in \Sigma_1 \cup \{q_n\}$.

It can be extended to words in $\Sigma^*$ and configurations in $\Sigma^{\mathbb{Z}}$.

▷ Claim 12. Let $c$ be a configuration in $\Sigma^Z$ and $i \in \mathbb{Z}$ a coordinate such that there exists $j < i < k$ with $c_j = \boxed{*} = c_k$. Then for any $t > s_b d_i$ (as in Claim 7),

$$\pi_2\left(\mathcal{F}_M^t(c)_i\right) \in \left\{\mathcal{F}_1^t(\phi(c))_i, q_n\right\}$$

We extend here $\pi_2$ as the identity to $\Sigma_1 \cup \{q_n\}$.

Proof. As $t > s_b d_i$, the cell at coordinate $i$ is in the protected area (above $\boxed{*}$ states or counters) at time $t$. Then the second layer has been computed with the rule of $\mathcal{F}_1$ and the second layer of the configuration rewritten by counters into images of $\phi(c)$. The only way to interrupt the computation of $\mathcal{F}_1$ is to erase the cell and write $q_n$, hence the claim. ◁

### 4.2.3 Proof of the theorem

It remains to prove the next 2 lemmas.

▶ **Lemma 13.** *If $M$ eventually halts on the empy input, then $\tilde{\omega}(\mathcal{F}_M) = \tilde{\omega}(\mathcal{F}_0) \in \mathcal{P}$.*

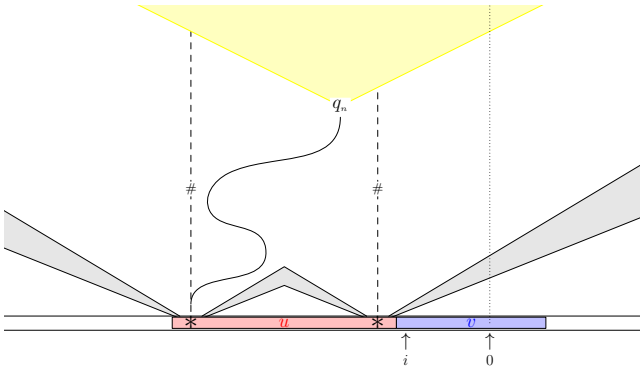



**Figure 7** The word $v$ (in blue) is supposed to enable state $s$. Then for a good choice of $u$ (in red), a segment will simulate a computation of $M$ that eventually halts and produces $q_n$. This state spreads (in yellow) and eventually reaches coordinate 0.

**Proof.** Suppose that $M$ eventually halts on the empty input. Then there exists a large enough size $S$ such that the computation in any segment larger than $S$ has enough time and space to reach its end. Then the state $q_n$ appears and spreads at speed 1 in both directions except if it encounters an inner or outer border of a counter.

If some state $s \in \Sigma$ occurs in $\tilde{\omega}(\mathcal{F}_M)$ then according to Lemma 6, there exists a word $v$ that enables it when placed at position $i \in \mathbb{Z}$. Take now $u = (\_\boxed{*}\_^{S}\boxed{*}\_, x_0^{S+4})$, $w$ the empty word and some $c \in [uvw]_{i-|u|}$. Counters are generated by the two $\boxed{*}$ states at coordinates $i-(S+3)$ and $i-2$, hence there exists $t_0 \in \mathbb{N}$ such that at time $t_0$, the cell 0 has been crossed by counters generated by $\boxed{*}$ states. According to Claim 7, it will not contain any state of outer or inner border of a counter anymore. Moreover, a segment is created between coordinates $i-(S+3)$ and $i-2$. As it is large enough, the state $q_n$ will be written at time $t_1 \in \mathbb{N}$. Then it will spread and reach cell 0 before time $t_1 + \max(|i-(S+2)|, |i-2|)$ or $t_0$ if the inner border of a counter slows it down. This is illustrated by Figure 7. Hence there exists $t_2 \in \mathbb{N}$ such that $\forall t \geq t_2, \mathcal{F}_M^t(c) \in [s] \Leftrightarrow s = q_n$. Thus $\tilde{\omega}(\mathcal{F}_M) \subseteq \{q_n\}^{\mathbb{Z}}$. As $\tilde{\omega}(\mathcal{F}_M)$ cannot be empty, we have $\tilde{\omega}(\mathcal{F}_M) = \{q_n\}^{\mathbb{Z}} = \tilde{\omega}(\mathcal{F}_0)$ and $\tilde{\omega}(\mathcal{F}_M) \in \mathcal{P}$. ◀

▶ **Lemma 14.** *If $M$ never halts on the empty input, then $\tilde{\omega}(\mathcal{F}_M) = \tilde{\omega}(\mathcal{F}_1) \notin \mathcal{P}$.*

**Proof.** Suppose now that $M$ never halts on the empty input. We will show that $\tilde{\omega}(\mathcal{F}_1) = \tilde{\omega}(\mathcal{F}_M)$.

First, we show that:

▷ Claim 15. $\tilde{\omega}(\mathcal{F}_M) \subseteq \tilde{\omega}(\mathcal{F}_1)$

Proof. Let $s$ be a word that occurs in $\tilde{\omega}(\mathcal{F}_M)$. According to Lemma 6, there exists a word $v$ that enables $s$ when placed at coordinate $i$. As any word containing $v$ as a factor also enables $s$, we can choose $v$ such that $i < 0$ and $i + |v| > |s|$.

We prove that $v' = \phi(v)$ at coordinate $i$ enables $s$ for $\mathcal{F}_1$. To do so, we will use Lemma 6. Take $u', w' \in \Sigma_1^*$ and denote $u = (\boxed{*}\_^{|u'|-1}, u')$ and $w = (\_^{|w'|-1}\boxed{*}, w')$. Denote $n = |uvw|$, $T \geq \max(s_b n, f(n) + n)$ (where $s_b$ is the speed of inner borders of counters), $z_1 = i - |u|$ and $z_2 = i + |vw|$. Apply Lemma 6 with $\mathcal{F}_M$, $v$, $u$ and $w$. For infinitely many times $t$, there exist a configuration $c \in [uvw]_{i-|u|}$ such that $\mathcal{F}_M^t(c) \in [s]$. Using Claim 12 with cells at coordinates $z_1$ and $z_2$ containing state $\boxed{*}$, we get that for any $t > T$,

$$\forall z_1 \leq j \leq z_2, \pi_2\left(\mathcal{F}_M^t(c)_j\right) \in \left\{\mathcal{F}_1^t(\phi(c))_j, q_n\right\}$$

That is : $\pi_2(s) = \pi_2\left(\mathcal{F}_M^t(c)_{[0,|u|-1]}\right) \in \left\{\mathcal{F}_1^t(\phi(c))_{[0,|u|-1]}, q_n^{|u|}\right\}$.

Due to the $\boxed{*}$ states placed at coordinates $z_1$ and $z_2$, we can also apply Claim 11 and we get that the computation is finished in any segment between coordinates $z_1$ and $z_2$ at time $f(n)$. After $n$ more steps, the potential abortion signals have reached the borders and every cell between coordinates $z_1$ and $z_2$ contains a state in $\Sigma_1 \cup \{q_n\}$. Moreover, as these cells belonged to a segment in the protected area, and since $M$ never halts on the empty input, this state cannot be $q_n$. Hence $s \in \Sigma^*$ and as $\pi_2$ is the identity on $\Sigma$, necessarily $s = \pi_2(s) = \mathcal{F}_1^t(\phi(c))_{[0,|u|-1]}$.

As $\phi(c) \in [u'v'w']_{i-|u'|}$ and as $\mathcal{F}_1^t(\phi(c))_{[0,|u|-1]} = s$ for infinitely many times $t$, Lemma 6 allows to conclude that $v'$ enables $s$ that is $v'$ occurs in $\tilde{\omega}(\mathcal{F}_1)$. ◁

Then we prove the opposite:

▷ Claim 16. $\tilde{\omega}(\mathcal{F}_1) \subseteq \tilde{\omega}(\mathcal{F}_M)$

Proof. Let $s$ be a word that occurs in $\tilde{\omega}(\mathcal{F}_1)$. According to Lemma 6, there exists a word $v$ that enables it when placed at coordinate $i$. We prove that $v' = (\_^{|i|}\boxed{*}^{|v|}\_^{|i|+|s|}, x_0^{|i|}vx_0^{|i|+|s|})$ at coordinate $i - |i|$ enables $s$ for $\mathcal{F}_M$.

For any $u', w' \in \Sigma^*$, denote $n = |u'v'w'|$. Let $T \geq \max(s_b n, f(n) + n)$ (where $s_b$ is still the speed of inner borders of counters) and denote

- $u = \phi(\pi_2(u'))x_0^{|i|}$;
- $w = x_0^{|i|+|s|}\phi(\pi_2(w'))$.

As $v$ enables $s$ for $\mathcal{F}_1$, there exists $c \in [uvw]_{i-|u|}$ and $t \geq T$ such that $\mathcal{F}_1^t(c) \in [s]$. We can write $c$ as $c_- uvwc_+$ where $c_-$ and $c_+$ are semi-infinite configuration in ${}^{\omega}\Sigma_1$ and $\Sigma_1^{\omega}$ respectively. Define $c' = ({}^{\omega}\boxed{*}, c_-)u'v'w'(\boxed{*}^{\omega}, c_+) \in [u'v'w']_{i-|i|-|u'|}$, we will prove that $\mathcal{F}_M^t(c') \in [s]$. First, note that $c = \phi(\pi_2(c'))$. Then using Claim 12, we have that for every $j \in [|i - |i|, i + |i| + |s|]$:

$$\pi_2\left(\mathcal{F}_M^t(c')_j\right) \in \left\{\mathcal{F}_1^t(c)_j, q_n\right\}$$

As $t \geq T \geq s_b n$ and $M$ does not halt on the empty input, $\pi_2\left(\mathcal{F}_M^t(c')\right)_j \neq q_n$. And as $t \geq T \geq f(n) + n$, the computation is aborted in every segment fully located between coordinates $|i - |i|$ and $i + |i| + |s|$ before step $f(n)$. After $n$ more steps, the first layer of these segments is erased, in particular for coordinates $j$ with $0 \leq j < |s|$. Hence $\mathcal{F}_M^t(c')_{[0,|s|-1]} = \pi_2\left(\mathcal{F}_M^t(c')\right)_{[0,|s|-1]} = \mathcal{F}_1^t(c)_{[0,|s|-1]} = s$ and $s \in \tilde{\omega}(\mathcal{F}_M)$. ◁

◀

The last two lemmas show that $M \mapsto \mathcal{F}_M$ is a reduction from the Halting problem of Turing machines on empty input to the problem of decision of $\mathcal{P}$.

## 5 Conclusion and perspectives

We proved Rice's theorem for generic limit sets of CA, which means that for example generic nilpotency is undecidable. In the case of limit sets and $\mu$-limit sets, the nilpotency problem has the lowest complexity in the arithmetical hierarchy amongst properties of limit or $\mu$-limit sets ($\Sigma_1^0$-complete for limit sets and $\Pi_3^0$-complete for $\mu$-limit sets). It may be the case once more for generic limit sets. Lemma 6 gives a $\Pi_3^0$ upper bound on the complexity of generic nilpotency and Törmä suggests in [12] that the exact complexity could be obtained using a construction close to the one presented in [1] or in the present paper. One might think that another version of Rice's theorem could be deduced where the lower bound of complexity on non trivial properties of generic limit sets is higher than $\Sigma_1^0$.

Using again constructions of [1], one can certainly prove properties similar to the ones obtained on $\mu$-limit sets in the same paper, but also build examples to show that the languages of $\mu$-limit set and generic limit set can have totally distinct complexities like $\Sigma_3^0$-complete versus a full-shift.

## References


1 Laurent Boyer, Martin Delacourt, Victor Poupet, Mathieu Sablik, and Guillaume Theyssier. $\mu$-limit sets of cellular automata from a computational complexity perspective. *Journal of Computer and System Sciences*, 81, September 2013. doi:10.1016/j.jcss.2015.05.004.

2 Benjamin Hellouin de Menibus and Mathieu Sablik. Characterisation of sets of limit measures after iteration of a cellular automaton on an initial measure. *CoRR*, abs/1301.1998, 2013. URL: http://arxiv.org/abs/1301.1998.

3 Martin Delacourt. Rice's theorem for $\mu$-limit sets of cellular automata. In *ICALP (2)*, pages 89–100, 2011. doi:10.1007/978-3-642-22012-8_6.

4 Saliha Djenaoui and Pierre Guillon. The generic limit set of cellular automata. working paper or preprint, 2018. URL: https://hal.archives-ouvertes.fr/hal-01861590.

5 Gustav Arnold Hedlund. Endomorphisms and automorphisms of the shift dynamical systems. *Mathematical Systems Theory*, 3(4):320–375, 1969. doi:10.1007/BF01691062.

6 Karel Culik II, Jan Pachl, and Sheng Yu. On the limit sets of cellular automata. *SIAM Journal on Computing*, 18(4):831–842, 1989.

7 Jarkko Kari. The nilpotency problem of one-dimensional cellular automata. *SIAM Journal on Computing*, 21:571–586, 1992. doi:10.1137/0221036.

8 Jarkko Kari. Rice's theorem for the limit sets of cellular automata. *Theoretical Computer Science*, 127:229–254, 1994. doi:10.1016/0304-3975(94)90041-8.

9 Jarkko Kari. Theory of cellular automata: A survey. *Theoretical Computer Science*, 334(1):3–33, 2005. doi:10.1016/j.tcs.2004.11.021.

10 Petr Kůrka and Alejandro Maass. Limit sets of cellular automata associated to probability measures. *Journal of Statistical Physics*, 100(5-6):1031–1047, 2000.

11 John Milnor. On the concept of attractor. *Communications in Mathematical Physics*, 99(2):177–195, 1985. doi:10.1007/BF01212280.

12 Ilkka Törmä. Complexity of generic limit sets of cellular automata. In Hector Zenil, editor, *Cellular Automata and Discrete Complex Systems*, pages 126–138, Cham, 2020. Springer International Publishing.

13 John von Neumann. *Theory of Self-Reproducing Automata*. University of Illinois Press, Champaign, IL, USA, 1966.

# Cellular Automata and Kan Extensions

**Alexandre Fernandez**
Université Paris Est Creteil, LACL, F-94010 Creteil, France

**Luidnel Maignan**
Université Paris Est Creteil, LACL, F-94010 Creteil, France

**Antoine Spicher**
Université Paris Est Creteil, LACL, F-94010 Creteil, France

## Abstract

In this paper, we formalize precisely the sense in which the application of a cellular automaton to partial configurations is a natural extension of its local transition function through the categorical notion of Kan extension. In fact, the two possible ways to do such an extension and the ingredients involved in their definition are related through Kan extensions in many ways. These relations provide additional links between computer science and category theory, and also give a new point of view on the famous Curtis-Hedlund theorem of cellular automata from the extended topological point of view provided by category theory. These links also allow to relatively easily generalize concepts pioneered by cellular automata to arbitrary kinds of possibly evolving spaces. No prior knowledge of category theory is assumed.

**2012 ACM Subject Classification** Theory of computation → Rewrite systems; Theory of computation → Models of computation

**Keywords and phrases** Cellular Automata, Kan Extension, Category Theory, Global Transformation

**Digital Object Identifier** 10.4230/OASIcs.AUTOMATA.2021.7

## 1 Introduction

Cellular automata are usually presented either as a local behavior extended to a global and uniform one or as a continuous uniform global behavior for the appropriate topology [1, 3]. We offer here a third, fruitful, point of view easing many generalizations of the concepts pioneered by cellular automata, *e.g.* via so-called global transformation [2, 5]. The goal of this paper is not to elaborate on these generalizations but to focus on some simple foundational bridges allowing these generalizations. In particular, we focus on Kan extensions, a categorical notion allowing, as we show here, to capture local/global descriptions [4]. While categories are generalizations of monoids and posets, the case of cellular automata can be fully treated in terms of posets only. Once the involved structures made clear via posets, the transition to categories is precisely what enables the generalizations in a surprisingly smooth way as discussed in the final section.

In this paper, we recall the direct definitions of cellular automata on groups, local transition function, global transition function, shift action, and also consider the counterparts of these functions on arbitrary partial configurations. This bigger picture allows to show that the various local/global relations between these objects are all captured by left and right Kan extensions, the latter providing an alternative definition of these objects. The proofs are provided in detail to show how the concept can be easily manipulated once understood. We also introduce slightly more generality than one would typically need in order to enrich the presentation of Kan extensions in a hopefully useful way. In the final section, we comment on the link with Curtis-Hedlund theorem and discuss briefly the smooth transition to more general systems where the space itself has to evolve.

27th IFIP WG 1.5 International Workshop on Cellular Automata and Discrete Complex Systems (AUTOMATA 2021).
Editors: Alonso Castillo-Ramirez, Pierre Guillon, and Kévin Perrot; Article No. 7; pp. 7:1–7:12

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

# 2 Cellular Automata and Kan Extensions

Let us give some basic definitions to fix the notations. We also note small caveats early on, to avoid having to deal with many unrelated details at the same time in a single proof or construction later on.

## 2.1 Cellular Automata

▶ **Definition 1.** *A* group *is a set $G$ with a binary operation $- \cdot - : G \times G \to G$ which is associative, which has a neutral element $1$ and for which any $g \in G$ has inverse $g^{-1}$. A right action of the group on a set $X$ is a binary operation $- \blacktriangleleft - : X \times G \to X$ such that $x \blacktriangleleft 1 = x$ and $(x \blacktriangleleft g) \blacktriangleleft h = x \blacktriangleleft (g \cdot h)$.*

In cellular automata, the group $G$ represents the space, each element $g \in G$ being at the same time an absolute and a relative position. This space is decorated with states that evolve through local interactions only. The classical formal definitions go as follows and work with the entire, often infinite, space.

▶ **Definition 2.** *A* cellular automaton *on a group $G$ is given by a finite* neighborhood *$N \subseteq G$, a finite set of* states *$Q$, and a* local transition function *$\delta : Q^N \to Q$. The elements of the set $Q^N$ are called* local configurations*. The elements of the set $Q^G$ are called* global configurations *and a right action $- \blacktriangleleft - : Q^G \times G \to Q^G$ is defined on $Q^G$ by $(c \blacktriangleleft g)(h) = c(g \cdot h)$. The* global transition function *$\Delta : Q^G \to Q^G$ of such a cellular automaton is defined as $\Delta(c)(g) = \delta((c \blacktriangleleft g) \restriction N)$.*

▶ **Proposition 3.** *The latter right action is indeed a right action.*

**Proof.** For any $g, h \in G$, we have $((c \blacktriangleleft g) \blacktriangleleft h)(i) = (c \blacktriangleleft g)(h \cdot i) = c(g \cdot h \cdot i) = (c \blacktriangleleft (g \cdot h))(i)$ for any $i \in G$, so $((c \blacktriangleleft g) \blacktriangleleft h) = (c \blacktriangleleft (g \cdot h))$ and also $(c \blacktriangleleft 1)(i) = c(1 \cdot i) = c(i)$ as required by Definition 1 of right actions. ◀

This choice of definition and right notation for the so called shift action has two advantages. Firstly, the definition of the action is a simple associativity. Secondly, when instantiated with $G = \mathbb{Z}$ with sum, the content of $c \blacktriangleleft 5$ is the content of $c$ shifted to the left, as the symbols indicates. Indeed, for $c' = c \blacktriangleleft 5$, $c'(-5) = c(0)$ and $c'(0) = c(5)$.

▶ **Proposition 4.** *For all $c \in Q^G$ and $g \in G$, $\Delta(c)(g)$ is determined by $c \restriction g \cdot N$.*

**Proof.** Indeed, $\Delta(c)(g) = \delta((c \blacktriangleleft g) \restriction N)$ so the value is determined by $(c \blacktriangleleft g) \restriction N$. But for any $n \in N$, $(c \blacktriangleleft g)(n) = c(g \cdot n)$ by definition of $\blacktriangleleft$. ◀

In common cellular automata terms, this proposition means that the *neighborhood of $g$* is $g \cdot N$, in this order. Let us informally call objects of the form $c \restriction g \cdot N \in \bigcup_{g \in G} Q^{g \cdot N}$ a *shifted local configuration*. Note that, at our level of generality, two different positions $g \neq g' \in G$ might have the same neighborhood $g \cdot N = g' \cdot N$. Although the injectivity of the function $(- \cdot N)$ could be a useful constraint to add, which is often verified in practice, we do not impose it so the reader should keep this in mind.

▶ **Proposition 5.** *The function $- \cdot N : G \to \mathbf{2}^G$ is not necessarily injective.*

**Proof.** Considering the group $G = \mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}$ and $N = \{(0,0), (1,0)\}$, we have $(0,0) + N = (1,0) + N = \{(0,0), (1,0)\}$ because of the torsion. ◀

Because of this, it is useful to replace the shifted local configurations, *i.e.* the union $\bigcup_{g\in G} Q^{g\cdot N}$, by the disjoint union $\bigcup_{g\in G}(\{g\}\times Q^{g\cdot N})$. The elements of the latter are of the form $(g\in G, c\in Q^{g\cdot N})$ and keep track of the considered "center" of the neighborhood. More explicitly, two elements $(g_0, c\restriction g_0\cdot N), (g_1, c\restriction g_1\cdot N)\in\bigcup_{g\in G}(\{g\}\times Q^{g\cdot N})$ are different as soon as $g_0\neq g_1$ even if $g_0\cdot N = g_1\cdot N$. This encodes things according to the intuition of a centered neighborhood.

## 2.2 The Poset of (Partial) Configurations

In the previous formal statements, one sees different kinds of configurations, explicitly or implicitly: global configurations $c\in Q^G$, local configurations $(c\blacktriangleleft g)\restriction N\in Q^N$, shifted local configurations $c\restriction g\cdot N\in Q^{g\cdot N}$, and their resulting "placed states" $(g\mapsto\Delta(c)(g))\in Q^{\{g\}}$. In the cellular automata literature, one often considers configurations defined on other subsets of the space, *e.g.* finite connected subsets. More generally, we are interested in all partial configurations $Q^S$ for arbitrary subsets $S\subseteq G$. The restriction operation $(-\restriction -)$ used many times above gives a partial ordering of these partial configurations.

▶ **Definition 6.** *A* (partial) configuration *$c$ is a partial function from $G$ to $Q$. Its domain of definition is denoted $|c|$ and called its* support*. The set of all configurations is denoted* $\mathrm{Conf} = \bigcup_{S\subseteq G} Q^S$*. We extend the previous right action $\blacktriangleleft$ and define it to map each $c\in\mathrm{Conf}$ to $c\blacktriangleleft g$ having support $|c\blacktriangleleft g| = \{h\in G\mid g\cdot h\in|c|\}$ and states $(c\blacktriangleleft g)(h) = c(g\cdot h)$.*

▶ **Proposition 7.** *The latter right action is well-defined and is a right action.*

**Proof.** The configuration $c\blacktriangleleft g$ is well-defined on all of its support. Indeed for any $h\in|c\blacktriangleleft g|$, $(c\blacktriangleleft g)(h) = c(g\cdot h)$ and $g\cdot h\in|c|$ by definition of $h$. The right action property is verified as in the proof of Proposition 3. ◀

Let us restate Proposition 4 more precisely using Definition 6.

▶ **Proposition 8.** *For all $c\in Q^G$ and $g\in G$, $(c\blacktriangleleft g)\restriction N = (c\restriction g\cdot N)\blacktriangleleft g$.*

**Proof.** Indeed, $|(c\restriction g\cdot N)\blacktriangleleft g| = \{h\in G|g\cdot h\in|(c\restriction g\cdot N)|\} = \{h\in G|g\cdot h\in g\cdot N\} = N = |(c\blacktriangleleft g)\restriction N|$. Also for any $n\in N$, $((c\blacktriangleleft g)\restriction N)(n) = (c\blacktriangleleft g)(n) = c(g\cdot n)$ and $((c\restriction g\cdot N)\blacktriangleleft g)(n) = (c\restriction g\cdot N)(g\cdot n) = c(g\cdot n)$. ◀

▶ **Definition 9.** *A* partial order *on a set $X$ is a binary relation $\preceq\subseteq X\times X$ which is reflexive, transitive, and antisymmetric. A set endowed with a partial order is called a* partially ordered set*, or* poset *for short.*

▶ **Definition 10.** *Given any two configurations $c, c'\in\mathrm{Conf}$, we set $c\preceq c'$ if and only if $\forall g\in|c|, g\in|c'|\wedge c(g) = c'(g)$. This is read "$c$ is a subconfiguration of $c'$" or "$c'$ is a superconfiguration of $c$".*

▶ **Proposition 11.** *The set* Conf *with this binary relation is a poset. In this poset, the shifted local configurations $c\in\bigcup_{g\in G} Q^{g\cdot N}$ are subconfigurations of the (appropriate) global configurations $c'\in Q^G$. Shifted local configurations form an antichain. Global configurations form an antichain.*

**Proof.** As can be readily seen, since each global configuration restricts to many shifted local configurations, and recalling that an antichain is a subset $S$ of the poset such that neither $x\preceq x'$ nor $x'\preceq x$ hold for any two different $x, x'\in S$. ◀

## 2.3 Kan Extensions (in the 2-Category of Posets)

Given three sets $A$, $B$ and $C$ such that $A \subseteq B$, we say that a function $g : B \to C$ extends a function $f : A \to C$ if $g \restriction A = f$, or equivalently if $g \circ i = f$ where $i$ is the obvious injective function from $A$ to $B$. For a given $f : A \to C$, there are typically many possible extensions. Roughly speaking, Kan extensions formalizes, among many things, the mathematical practice where extensions are rarely arbitrary. One usually chooses the "best" or "most natural" extensions. There is therefore an implicit comparison considered between the extensions.

This is the reason why Kan extensions are formally defined at the level of 2-categories: $A$, $B$, $C$ are objects, $f$, $g$, $i$, and all (not necessarily "most natural") extensions are 1-morphisms between these objects, and the "naturality" comparison between 1-morphisms are 2-morphisms. However, we do not need to discuss things at this level of generality here. For our particular case, the objects are posets, the 1-morphisms are monotonic functions and the monotonic functions are compared pointwise.

▶ **Definition 12.** *Given two posets $(X, \preceq_X)$ and $(Y, \preceq_Y)$, a function $f : X \to Y$ is said to be* monotonic *if for all $x, x' \in X$, $x \preceq_X x'$ implies $f(x) \preceq_Y f(x')$.*

▶ **Proposition 13.** *For any $g \in G$, the function $(- \blacktriangleleft g) : \mathrm{Conf} \to \mathrm{Conf}$ is monotonic.*

**Proof.** Given any $c, c' \in \mathrm{Conf}$ such that $c \preceq c'$, this claim is equivalent to:

$$\begin{aligned} & (c \blacktriangleleft g) \preceq (c' \blacktriangleleft g) \text{ (by Def 12)} \\ \iff & \forall h \in |c \blacktriangleleft g|; h \in |c' \blacktriangleleft g| \wedge (c \blacktriangleleft g)(h) = (c' \blacktriangleleft g)(h) \text{ (Def 10)} \\ \iff & \forall h \in G \text{ s.t. } g \cdot h \in |c|; g \cdot h \in |c'| \wedge c(g \cdot h) = c'(g \cdot h) \text{ (Def 6)} \end{aligned}$$

which is true by the application of Definition 10 of $c \preceq c'$ on $g \cdot h$. ◀

▶ **Definition 14.** *Given two posets $(X, \preceq_X)$ and $(Y, \preceq_Y)$, we define the binary relation $- \Rightarrow -$ on the set of all monotonic functions from $X$ to $Y$ by $f \Rightarrow f' \iff \forall c \in X, f(c) \preceq_Y f'(c)$.*

▶ **Proposition 15.** *Given two posets $(X, \preceq_X)$ and $(Y, \preceq_Y)$, the set of monotonic functions between them together with this binary relation forms a poset.*

**Proof.** As one can easily check. ◀

▶ **Definition 16.** *In this setting, given three posets $A$, $B$ and $C$, and three monotonic functions $i : A \to B$, $f : A \to C$ and $g : B \to C$, $g$ is said to be the* left (resp. right) Kan extension of $f$ along $i$ if $g$ *is the $\Rightarrow$-minimum (resp. $\Rightarrow$-maximum) element in the set of monotonic functions $\{h : B \to C \mid f \Rightarrow h \circ i\}$ (resp. $\{h : B \to C \mid h \circ i \Rightarrow f\}$).*

This concept is particularly useful because, whenever it applies, it is also a complete characterization as stated in the following proposition in the left case.

▶ **Proposition 17.** *The left (resp. right) Kan extension $g$ is unique when it exists.*

**Proof.** It is defined as the minimum of a set, and as any minimum, it may not exist, but when it does, it is always unique. ◀

Another suggestive way to read the concept of Kan extensions with respect to this paper is to say that a function $g$ on a poset can be summarized into, or generated by, a part of its behavior $f$ on just a small part of the poset. Note however that $i$ does not need to be injective in this definition.

# 3 Kan Extensions in Cellular Automata

## 3.1 A First Approach To Partial Configurations

The first, intuitive, approach is to take a configuration $c$, look for all places $g$ where the whole neighborhood $g \cdot N$ is defined and to take the local transition result of these places only. We first give a direct formal definition, and then show that this is a left Kan extension. This shows in particular that the global transition function is the left Kan extension of the "fully shifted" local transition. The sense of "fully shifted" is described below and is only necessary because we restrict ourselves to posets, as discussed in the final section of this paper.

### 3.1.1 A Direct Definition

▶ **Definition 18.** *The* interior *of a subset $S \subseteq G$ is* $\mathrm{int}(S) = \{g \in G \mid g \cdot N \subseteq S\}$.

▶ **Definition 19.** *The* coarse transition function $\Phi : \mathrm{Conf} \to \mathrm{Conf}$ *is defined for all $c \in \mathrm{Conf}$ as $|\Phi(c)| = \mathrm{int}(|c|)$ and $\Phi(c)(g) = \delta((c \blacktriangleleft g) \restriction N)$.*

▶ **Proposition 20.** *For any $c \in \mathrm{Conf}$ and $g \in G$, the statements $g \in \mathrm{int}(|c|)$, $g \cdot N \subseteq |c|$, and $N \subseteq |c \blacktriangleleft g|$ are equivalent. (So $\Phi$ is well-defined in Definition 19.)*

**Proof.** The first and second statements are equivalent by Definition 18 of int. The second and third statements are equivalent by Definition 6 of $\blacktriangleleft$. ◀

Remember Proposition 5. If we do have injectivity of neighborhoods, we have $\mathrm{int}(g \cdot N) = \{g\}$. But since we do not assume it, we only have the following.

▶ **Proposition 21.** *Let $S \subseteq G$. It is always the case that $S \subseteq \mathrm{int}(S \cdot N)$ but we do not necessarily have equality, even when $S = \{g\}$ for some $g \in G$.*

**Proof.** Consider any $s \in S$. Clearly, $s \cdot N \subseteq S \cdot N$, so by Definition 18, $s \in \mathrm{int}(S \cdot N)$. However, we do not have equality in the example of the proof of Proposition 5 with $S = \{(0,0)\}$. Indeed, in this case, $\mathrm{int}(S \cdot N) = \{(0,0),(1,0)\}$. ◀

Another useful remark on which we come back below is the following.

▶ **Proposition 22.** *For any $g \in G$, any $M \subsetneq N$, and any $c \in Q^{g \cdot M}$, $|\Phi(c)| = \emptyset$. Also, for any $c \in Q^G$, $|\Phi(c)| = |\Delta(c)|$.*

**Proof.** By Definition 19 of $\Phi$. ◀

### 3.1.2 Characterization as a Left Kan Extension

The coarse transition function $\Phi$ is defined on the set of all configurations Conf and we claim that it is generated, in the Kan extension sense, by the local transition function $\delta$ shifted everywhere. We define the latter, with Proposition 5 in mind.

▶ **Definition 23.** *We define* Loc *to be the poset* $\mathrm{Loc} = \bigcup_{g \in G}(\{g\} \times Q^{g \cdot N})$ *with trivial partial order $(g,c) \preceq (g',c') \iff (g,c) = (g',c')$. The "fully shifted local transition function" $\overline{\delta} :$ Loc $\to$ Conf is defined, for any $(g,c) \in$ Loc as $|\overline{\delta}(g,c)| = \{g\}$ and $\overline{\delta}(g,c)(g) = \delta(c \blacktriangleleft g)$. The second projection of* Loc *is the monotonic function $\pi_2 :$ Loc $\to$ Conf defined as $\pi_2(g,c) = c$.*

▶ **Proposition 24.** Loc *is a poset and $\overline{\delta}$ and $\pi_2$ are monotonic functions.*

**Proof.** Indeed, the identity relation is an order relation and any function respects the identity relation. ◀

▶ **Proposition 25.** *The coarse transition function* $\Phi$ *is monotonic.*

**Proof.** Indeed, take $c, c' \in \mathrm{Conf}$ such that $c \preceq c'$. We want to prove that $\Phi(c) \preceq \Phi(c')$ and this is equivalent to:

$$\begin{aligned}
&\forall g \in |\Phi(c)|, g \in |\Phi(c')| \wedge \Phi(c)(g) = \Phi(c')(g) \\
\iff &\forall g \in \mathrm{int}(|c|), g \in \mathrm{int}(|c'|) \wedge \delta(c \blacktriangleleft g \upharpoonright N) = \delta(c' \blacktriangleleft g \upharpoonright N) \\
\iff &\forall g \in G \text{ s.t. } g \cdot N \subseteq |c|, g \cdot N \subseteq |c'| \wedge \delta(c \blacktriangleleft g \upharpoonright N) = \delta(c' \blacktriangleleft g \upharpoonright N),
\end{aligned}$$

by Definition 6 of $\preceq$, Definition 19 of $\Phi$, and Definition 18 of int. The final statement is implied by:

$$\begin{aligned}
&\forall g \in G \text{ s.t. } g \cdot N \subseteq |c|, g \cdot N \subseteq |c'| \wedge c \blacktriangleleft g \upharpoonright N = c' \blacktriangleleft g \upharpoonright N \\
\iff &\forall g \in G \text{ s.t. } g \cdot N \subseteq |c|, g \cdot N \subseteq |c'| \wedge \forall n \in N, (c \blacktriangleleft g)(n) = (c' \blacktriangleleft g)(n) \\
\iff &\forall g \in G \text{ s.t. } g \cdot N \subseteq |c|, g \cdot N \subseteq |c'| \wedge \forall n \in N, c(g \cdot n) = c'(g \cdot n),
\end{aligned}$$

the last equivalence being by Definition 6. To prove it, take $g \in G$ such that $g \cdot N \subseteq |c|$, and $n \in N$. By Definition 10, since $c \preceq c'$ and $g \cdot n \in |c|$, we have $g \cdot n \in |c'|$, and $c(g \cdot n) = c'(g \cdot n)$ as wanted. ◀

▶ **Proposition 26.** $\Phi$ *is the left Kan extension of* $\overline{\delta}$ *along* $\pi_2 : \mathrm{Loc} \to \mathrm{Conf}$.

**Proof.** By Definition 16 of left Kan extensions, we need to prove firstly that $\Phi$ is such that $\overline{\delta} \Rightarrow \Phi \circ \pi_2$, and secondly that it is smaller than any other such monotonic function.

For the first part, $\overline{\delta} \Rightarrow \Phi \circ \pi_2$ is equivalent to:

$$\begin{aligned}
&\forall (g, c) \in \mathrm{Loc}, \overline{\delta}(g, c) \preceq \Phi(c) \text{ (Defs. 14 and 23 of } \Rightarrow \text{ and } \pi_2) \\
\iff &\forall (g, c) \in \mathrm{Loc}, \forall h \in |\overline{\delta}(g, c)|, h \in |\Phi(c)| \wedge \overline{\delta}(g, c)(h) = \Phi(c)(h) \text{ (D. 10 of } \preceq) \\
\iff &\forall (g, c) \in \mathrm{Loc}, g \in |\Phi(c)| \wedge \delta(c \blacktriangleleft g) = \Phi(c)(g) \text{ (Def. 23 of } \overline{\delta}) \\
\iff &\forall (g, c) \in \mathrm{Loc}, g \cdot N \in |c| \wedge \delta(c \blacktriangleleft g) = \delta((c \blacktriangleleft g) \upharpoonright N) \text{ (Defs 18, 19 of } \Phi).
\end{aligned}$$

This last statement is true by Definition 23 of Loc, *i.e.* since $c \in Q^{g \cdot N}$, $c \blacktriangleleft g = (c \blacktriangleleft g) \upharpoonright N$.

For the second part, let $F : \mathrm{Conf} \to \mathrm{Conf}$ be a monotonic function such that $\overline{\delta} \Rightarrow F \circ \pi_2$. We want to show that $\Phi \Rightarrow F$, which is equivalent to:

$$\begin{aligned}
&\forall c \in \mathrm{Conf}, \Phi(c) \preceq F(c) \text{ (Def. 14 of } \Rightarrow) \\
\iff &\forall c \in \mathrm{Conf}, \forall g \in |\Phi(c)|, g \in |F(c)| \wedge \Phi(c)(g) = F(c)(g) \text{ (Def. 10 of } \preceq) \\
\iff &\forall c \in \mathrm{Conf}, \forall g \in \mathrm{int}(|c|), g \in |F(c)| \wedge F(c)(g) = \delta((c \blacktriangleleft g) \upharpoonright N) \text{ (Def. 19)}
\end{aligned}$$

So take $c \in \mathrm{Conf}$ and $g \in \mathrm{int}(|c|)$, and consider $d_g = c \upharpoonright g \cdot N$. Since $d_g \preceq c$ and $F$ is monotonic, we have $F(d_g) \preceq F(c)$. Moreover $\overline{\delta} \Rightarrow F \circ \pi_2$ and $(g, d_g) \in \{g\} \times Q^{g \cdot N} \subseteq \mathrm{Loc}$, so $\overline{\delta}(g, d_g) \preceq F(d_g)$ by Definitions 14 and 23 of $\Rightarrow$ and $\pi_2$. By transitivity $\overline{\delta}(g, d_g) \preceq F(c)$. By Definition 23 of $\overline{\delta}$ and Definition 10 of $\preceq$, we obtain $g \in |F(c)|$, and $F(c)(g) = \overline{\delta}(g, d_g)(g) = \delta((c \blacktriangleleft g) \upharpoonright N)$, as wanted. ◀

As a sidenote, remark that in order to have the equality $\overline{\delta} = \Phi \circ \pi_2$, one needs to have the injectivity of neighborhood function. Indeed, without injectivity, we have two different $g, g' \in G$ having the same neighborhood $M$, *i.e.* $M = g \cdot N = g' \cdot N$. This means that,

given any local configuration $c \in Q^M$ on this neighborhood, each pair $(g, c), (g', c) \in \mathrm{Loc}$ have different results $\overline{\delta}(g, c) \in Q^{\{g\}}$ and $\overline{\delta}(g', c) \in Q^{\{g'\}}$ with different support $\{g\}$ and $\{g'\}$. However, their common projection $\pi_2(g, c) = \pi_2(g', c) = c$ have a unique result $\Phi(c)$ with a support such that $\{g, g'\} \subseteq |\Phi(c)|$. So we have a strict comparison $\overline{\delta} \Rightarrow \Phi \circ \pi_2$. When the neighborhood function is injective, $\pi_2$ is also injective and the previous situation can not occur so we have equality $\overline{\delta} = \Phi \circ \pi_2$.

## 3.2 A Second Approach To Partial Configurations

For some applications, the previous definitions are too naive. For example, it is common to consider two cellular automata to be essentially the same if they generate the same global transition functions. However, here, two such cellular automata give different coarse transition function if they have a different neighborhood.

To refine the previous definitions, a second approach is to take a configuration $c$, and look at all places for which the result is already determined by the partial data defined in $c$. So we consider all $g \in G$ for which all completions of the data present on the defined neighborhood $g \cdot N \cap |c|$ into a configuration on the complete neighborhood $g \cdot N$ always lead to the same result by $\delta$.

### 3.2.1 A Direct Definition

▶ **Definition 27.** *For any $c \in \mathrm{Conf}$ and $g \in G$, let $c_g = c \restriction (g \cdot N \cap |c|)$.*

▶ **Definition 28.** *Given a configuration $c \in \mathrm{Conf}$, its* determined subset *is $\det(c) = \{g \in G \mid \exists q \in Q, \forall c' \in Q^{g \cdot N}, c' \restriction |c_g| = c_g \implies \delta(c' \blacktriangleleft g) = q\}$. For any $g \in \det(c)$, we denote $q_{c,g} \in Q$ the unique state $q$ having the mentioned property.*

Note that this definition depends on the cellular automaton local transition function $\delta$ and on the data of the configuration $c$, contrary to Definition 18 of interior that only depends on its neighborhood $N$ and on the support of the configuration.

▶ **Definition 29.** *Given a cellular automaton, its* fine transition function $\phi : \mathrm{Conf} \to \mathrm{Conf}$ *is defined as $|\phi(c)| = \det(c)$ and $\phi(c)(g) = q_{c,g}$,* i.e. *$\phi(c)(g) = \delta(c' \blacktriangleleft g)$ for any $c' \in Q^{g \cdot N}$ such that $c' \restriction |c_g| = c_g$.*

▶ **Proposition 30.** *The fine transition function $\phi$ is well defined.*

**Proof.** This is the case precisely because we restrict the support of $\phi(c)$ to the determined subset of the $c$. ◀

▶ **Proposition 31.** *Consider the constant cellular automaton $\delta(c) = q \; \forall c \in Q^N$ for a specific $q \in Q$ and regardless of the neighborhood $N$ chosen to represent it. We have $|\phi(c)| = G$ for any $c \in \mathrm{Conf}$.*

**Proof.** Indeed, even with no data at all, *i.e.* for $c$ such that $|c| = \emptyset$, the result at all position is determined and is $q$. ◀

Note that, contrary to Proposition 22 of the coarse transition function, the fine transition function definition is explicitly about considering non-empty results for some configurations of support $M \subsetneq N$. When there is no such "sub-local" configuration with determined result, the two transition functions are actually equal. But let us note insist on this point.

### 3.2.2 Characterization as a Right Kan Extension

As for the coarse transition function, the fine transition function $\phi$ is defined on the set of all configurations Conf and we claim that it is generated, in the Kan extension sense. We consider two ways to generate it and start by the simplest one. The second one is considered in the following section using sub-local configurations in order to be closer to the direct definition and to be a "from local to global" characterization.

▶ **Proposition 32.** *For any $g \in G$, the function $-_g : \text{Conf} \to \text{Conf}$ of Definition 27 is monotonic.*

**Proof.** As one can easily check. ◀

▶ **Proposition 33.** *The fine transition function $\phi$ is monotonic.*

**Proof.** Indeed, take $c_0, c_1 \in \text{Conf}$ such that $c_0 \preceq c_1$. We want to prove that $\phi(c_0) \preceq \phi(c_1)$ and this is equivalent to:

$$\begin{aligned} & \forall g \in |\phi(c_0)|, g \in |\phi(c_1)| \wedge \phi(c_0)(g) = \phi(c_1)(g) \text{ (Def 10 of } \preceq) \\ \iff & \forall g \in \det(c_0), g \in \det(c_1) \wedge q_{c_0,g} = q_{c_1,g} \text{ (Def 29 of } \phi) \end{aligned}$$

Take $g \in \det(c_0)$. We want to prove that $g \in \det(c_1)$, which means by Definition 28 of $\det(c_1)$:

$$\exists q \in Q, \forall c_2 \in Q^{g \cdot N}, c_2 \upharpoonright |(c_1)_g| = (c_1)_g \implies \delta(c_2 \blacktriangleleft g) = q$$

We claim that the property is verified with $q = q_{c_0,g}$. Indeed, take any $c_2 \in Q^{g \cdot N}$ such that $c_2 \upharpoonright |(c_1)_g| = (c_1)_g$. We also have that $c_2 \upharpoonright |(c_0)_g| = (c_0)_g$ since the hypothesis $c_0 \preceq c_1$ implies $(c_0)_g \preceq (c_1)_g$ by Proposition 32. By Definition 28 of $\det(c_0)$, we obtain that $\delta(c_2 \blacktriangleleft g) = q_{c_0,g}$, so $q = q_{c_0,g}$ has the wanted property, which implies that $g \in \det(c_1)$ as wanted. But the above property of $q$ set it to be precisely what we denote by $q_{c_1,g}$ (Def 28 of $q_{c_1,g}$), so $q_{c_0,g} = q_{c_1,g}$. ◀

▶ **Proposition 34.** *The fine transition function $\phi$ is the right Kan extension of the global transition function $\Delta$ along the inclusion $i : Q^G \to \text{Conf}$.*

**Proof.** By Definition 16 of right Kan extensions, we need to prove firstly that $\phi$ is such that $\phi \circ i \Rightarrow \Delta$, and secondly that it is greater than any other such monotonic functions.

For the first part, we actually have $\phi \circ i = \Delta$ since for any $c \in Q^G$, $|\phi(c)| = \det(c) = G = |\Delta(c)|$ and for any $g \in G$, we have $\phi(c)(g) = q_{c,g} = \delta(c_g \blacktriangleleft g) = \delta((c \upharpoonright g \cdot N) \blacktriangleleft g) = \delta((c \blacktriangleleft g) \upharpoonright N) = \Delta(c)(g)$ by Defs. 29, 28, 27, 2 of $\phi$, det, $c_g$ and $\Delta$ and Prop. 8.

For the second part, let $f : \text{Conf} \to \text{Conf}$ be a monotonic function such that $f \circ i \Rightarrow \Delta$. We want to show that $f \Rightarrow \phi$, which is equivalent to:

$$\begin{aligned} & \forall c \in \text{Conf}, f(c) \preceq \phi(c) \text{ (Def. 14 of } \Rightarrow) \\ \iff & \forall c \in \text{Conf}, \forall g \in |f(c)|, g \in |\phi(c)| \wedge f(c)(g) = \phi(c)(g) \text{ (Def. 10 of } \preceq) \\ \iff & \forall c \in \text{Conf}, \forall g \in |f(c)|, g \in \det(c) \wedge f(c)(g) = q_{c,g} \text{ (Def. 36 of } \phi) \\ \iff & \forall c \in \text{Conf}, \forall g \in |f(c)|, \forall c' \in Q^{g \cdot N}, c \preceq c' \implies f(c)(g) = \delta(c' \blacktriangleleft g) \text{ (D. 28)} \end{aligned}$$

So take $c \in \text{Conf}$ and $g \in |f(c)|$ and $c' \in Q^{g \cdot N}$ such that $c \preceq c'$. Consider any $c'' \in Q^G$ such that $c' \preceq c''$ (or equivalently $c'' \upharpoonright g \cdot N = c'$). Since $f$ is monotonic, we have $f(c) \preceq f(c'')$, which means that $f(c)(g) = f(c'')(g)$ by Def. 10. But since $f \circ i \Rightarrow \Delta$, we have $f(c)(g) = \Delta(c'')(g) = \delta((c'' \blacktriangleleft g) \upharpoonright N)$ by Def. 14 of $\Rightarrow$ and Def. 2 of $\Delta$. But by Prop. 8, $(c'' \blacktriangleleft g) \upharpoonright N = (c'' \upharpoonright g \cdot N) \blacktriangleleft g = c' \blacktriangleleft g$. ◀

▶ **Proposition 35.** *Let us consider another cellular automaton having neighborhood $N' \subseteq G$ and local transition function $\delta' : Q^{N'} \to Q$. Consider its corresponding global transition function $\Delta' : Q^N \to Q^N$ and fine transition function $\phi' : \mathrm{Conf} \to \mathrm{Conf}$. Then if $\Delta' = \Delta$, then $\phi' = \phi$.*

**Proof.** By Propositions 34 and 17, $\phi$ is determined by $\Delta$, and $\phi'$ by $\Delta'$. So $\Delta' = \Delta$ gives $\phi = \phi'$. ◀

## 3.3 Introducing Sub-Local Configurations

The direct definition of the fine transition function is explicitly about assigning a result for a configuration $c$ at a given $g \in G$ even when the whole neighborhood $g \cdot N$ is not complete. By isolating these "shifted sub-local configurations" in the poset of configurations, we can (right-)extend the local transition to them and show that, in the same way as the coarse transition function is the left Kan extension of the local transition function, the fine transition function is the left Kan extension of the sub-local transition function.

### 3.3.1 Direct Definition

▶ **Definition 36.** *We define* $\mathrm{Sub} = \bigcup_{g \in G, M \subseteq N} (\{g\} \times Q^{g \cdot M})$ *with partial order defined as* $(g, c) \preceq (g', c')$ *if and only if* $g = g'$ *and* $c \preceq c'$. *The "fully shifted sub-local transition function"* $\underline{\delta} : \mathrm{Sub} \to \mathrm{Conf}$ *is defined, for any* $g \in G$, *any* $M \subseteq N$ *and any* $c \in Q^{g \cdot M}$, *as* $|\underline{\delta}(g, c)| = \{g\} \cap \det(c)$ *and, if* $g \in \det(c)$, $\underline{\delta}(g, c)(g) = q_{c',g}$, i.e. $\underline{\delta}(g, c)(g) = \delta(c' \blacktriangleleft g)$ *for any* $c' \in Q^{g \cdot N}$ *such that* $c = c' \restriction |c|$. *The second projection of* Sub *is the function* $\pi_2 : \mathrm{Sub} \to \mathrm{Conf}$ *defined as* $\pi_2(g, c) = c$.

In this definition, a given sub-local configuration can result either in an empty configuration when the transition is not determined, or in a configuration with only singleton support when the transition is determined.

Note that for a given cellular automaton, it is possible to restrict the poset Sub to an antichain. Indeed, any time a result is determined by a sub-local configuration $(g, c)$, all bigger sub-local configuration $(g, c')$ with $c \preceq c'$ does not contribute anything new. We do not elaborate on this because this antichain would be different for each cellular automaton, blurring the global picture presented below.

### 3.3.2 Characterization as a Right Kan Extension

▶ **Proposition 37.** *The fully shifted sub-local transition function* $\underline{\delta}$ *is monotonic*

**Proof.** As usual, take $(g, c), (g', c') \in \mathrm{Sub}$ such that $(g, c) \preceq (g', c')$. First note that $g = g'$ by Definition 36. We want to prove that $\underline{\delta}(g, c) \preceq \underline{\delta}(g, c')$ and this is equivalent to:

$$\begin{aligned} & \forall h \in |\underline{\delta}(c)|, h \in |\underline{\delta}(c')| \wedge \underline{\delta}(c)(h) = \underline{\delta}(c')(h) \\ \iff & \forall h \in \{g\} \cap \det(c), h \in \{g\} \cap \det(c') \wedge q_{c,g} = q_{c',g} \\ \iff & g \in \det(c) \implies g \in \det(c') \wedge q_{c,g} = q_{c',g'}, \end{aligned}$$

by Definition 6 of $\preceq$ and Definition 36 of $\underline{\delta}$. The end of this proof is similar to the one of Proposition 33. ◀

▶ **Proposition 38.** *The fully shifted sub-local transition function* $\underline{\delta}$ *is the right Kan extension of the fully shifted local transition function* $\overline{\delta}$ *along the inclusion* $i : \mathrm{Loc} \to \mathrm{Sub}$.

**Proof.** By Definition 16 of right Kan extensions, we need to prove firstly that $\underline{\delta}$ is such that $\underline{\delta} \circ i \Rightarrow \overline{\delta}$, and secondly that it is greater than any other such monotonic functions.

For the first part, $\underline{\delta} \circ i \Rightarrow \overline{\delta}$ is equivalent to:

$$\begin{aligned}
&\forall (g,c) \in \mathrm{Loc}, \underline{\delta}(g,c) \preceq \overline{\delta}(g,c) \text{ (Def. 14 of } \Rightarrow) \\
\iff &\forall (g,c) \in \mathrm{Loc}, \forall h \in |\underline{\delta}(g,c)|, h \in |\overline{\delta}(g,c)| \wedge \underline{\delta}(g,c)(h) = \overline{\delta}(g,c)(h) \text{ (D. 10 } \preceq) \\
\iff &\forall (g,c) \in \mathrm{Loc}, g \in \det(c) \implies g \in |\overline{\delta}(g,c)| \wedge q_{c,g} = \overline{\delta}(g,c)(g) \text{ (Def. 36 of } \underline{\delta}) \\
\iff &\forall (g,c) \in \mathrm{Loc}, g \in \det(c) \implies g \in \{g\} \wedge q_{c,g} = \delta(c \blacktriangleleft g) \text{ (Def 23 of } \overline{\delta}).
\end{aligned}$$

This last statement is true by Def. 28 of $q_{c,g}$.

For the second part, let $f : \mathrm{Sub} \to \mathrm{Conf}$ be a monotonic function such that $f \circ i \Rightarrow \overline{\delta}$. We want to show that $f \Rightarrow \underline{\delta}$, which is equivalent to:

$$\begin{aligned}
&\forall (g,c) \in \mathrm{Sub}, f(g,c) \preceq \underline{\delta}(g,c) \text{ (Def. 14 of } \Rightarrow) \\
\iff &\forall (g,c) \in \mathrm{Sub}, \forall h \in |f(g,c)|, h \in |\underline{\delta}(g,c)| \wedge f(g,c)(h) = \underline{\delta}(g,c)(h) \text{ (Def. 10)} \\
\iff &\forall (g,c) \in \mathrm{Sub}, \forall h \in |f(g,c)|, h \in \{g\} \cap \det(c) \wedge f(g,c)(h) = q_{c,g} \text{ (Def. 36)}
\end{aligned}$$

So take $(g,c) \in \mathrm{Sub}$ and $h \in |f(g,c)|$. Consider any $c' \in \mathrm{Loc}$ such that $c \preceq c'$. Since $f$ is monotonic, we have $f(g,c) \preceq f(g,c')$, which means that $h \in |f(g,c')|$ and $f(g,c)(h) = f(g,c')(h)$ by Def. 10. But since $f \circ i \Rightarrow \overline{\delta}$, we have $h \in |\overline{\delta}(g,c')| = \{g\}$ and $f(g,c')(h) = \overline{\delta}(g,c')(h) = \delta(c' \blacktriangleleft g)$ by Def. 14 of $\Rightarrow$ and Def. 23 of $\overline{\delta}$. Since this is true for any $c'$, this establishes exactly the defining property of $\det(c)$ by Def. 28. ◀

### 3.3.3 The Second Approach as a Left Kan Extension

▶ **Proposition 39.** *The projection function $\pi_2 : \mathrm{Sub} \to \mathrm{Conf}$ is monotonic.*

**Proof.** As can be readily checked in Definition 36. ◀

▶ **Proposition 40.** *$\phi$ is the left Kan extension of $\underline{\delta}$ along $\pi_2 : \mathrm{Sub} \to \mathrm{Conf}$.*

**Proof.** By Definition 16 of left Kan extensions, we need to prove firstly that $\phi$ is such that $\underline{\delta} \Rightarrow \phi \circ \pi_2$, and secondly that it is smaller than any other such monotonic functions.

For the first part, $\underline{\delta} \Rightarrow \phi \circ \pi_2$ is equivalent to:

$$\begin{aligned}
&\forall (g,c) \in \mathrm{Sub}, \underline{\delta}(g,c) \preceq \phi(c) \text{ (Defs. 14 and 36 of } \Rightarrow \text{ and } \pi_2) \\
\iff &\forall (g,c) \in \mathrm{Sub}, \forall h \in |\underline{\delta}(g,c)|, h \in |\phi(c)| \wedge \underline{\delta}(g,c)(h) = \phi(c)(h) \text{ (Def 10 of } \preceq) \\
\iff &\forall (g,c) \in \mathrm{Sub}, g \in \det(c) \implies g \in |\phi(c)| \wedge q_{c,g} = \phi(c)(g) \text{ (Def. 36 of } \underline{\delta}) \\
\iff &\forall (g,c) \in \mathrm{Sub}, g \in \det(c) \implies g \in \det(c) \wedge q_{c,g} = q_{c,g} \text{ (Def 29 of } \phi),
\end{aligned}$$

a most trivial statement.

For the second part, let $f : \mathrm{Conf} \to \mathrm{Conf}$ be a monotonic function such that $\underline{\delta} \Rightarrow f \circ \pi_2$. We want to show that $\phi \Rightarrow f$, which is equivalent to:

$$\begin{aligned}
&\forall c \in \mathrm{Conf}, \phi(c) \preceq f(c) \text{ (Def. 14 of } \Rightarrow) \\
\iff &\forall c \in \mathrm{Conf}, \forall g \in |\phi(c)|, g \in |f(c)| \wedge \phi(c)(g) = f(c)(g) \text{ (Def. 10 of } \preceq) \\
\iff &\forall c \in \mathrm{Conf}, \forall g \in \det(c), g \in |f(c)| \wedge q_{c,g} = f(c)(g) \text{ (Def. 29 of } \phi)
\end{aligned}$$

So take $c \in \mathrm{Conf}$ and $g \in \det(c)$. Since $c_g \preceq c$ (Def 27) and $f$ is monotonic, we have $f(c_g) \preceq f(c)$. Moreover $\underline{\delta} \Rightarrow f \circ \pi_2$ and $(g, c_g) \in \mathrm{Sub}$ so $\underline{\delta}(g, c_g) \preceq f(c_g)$ by Definitions 14 and 23 of $\Rightarrow$ and $\pi_2$. By transitivity $\underline{\delta}(g, c_g) \preceq f(c)$. By Definition 36 of $\underline{\delta}$ and Definition 10 of $\preceq$, we therefore have $g \in |f(c)|$, and $f(c)(g) = \underline{\delta}(g, d_g)(g) = q_{c,g}$ as wanted. ◀

## 4 Final Discussion

There are additional simple structural facts to note about the monotonic functions considered. The first one is that the shift action on partial configurations, as given in Definition 6, is the right Kan extension of the shift action on global configurations, as given in Definition 2. Another one is that $\Phi \Rightarrow \phi$, hence the names of these transition functions, coarse and fine. In fact, any monotonic function $f : \mathrm{Conf} \to \mathrm{Conf}$ such that $\overline{\delta} \Rightarrow f \circ \pi_2$ is necessarily such that $\Phi \Rightarrow f \Rightarrow \phi$. This shows, in some sense, the efficiency of the simple constraints of monotonicity and $\overline{\delta} \Rightarrow f \circ \pi_2$.

In the formal development presented here, we explicitly "copy" a single local behavior $\delta$ on all $g \in G$ to obtain $\overline{\delta}$ and work with it. It is readily possible to put a different behavior on each $g \in G$, with no real modification to the proofs. The statements are therefore valid for non-uniform cellular automata and automata networks. We do not even insist of the number of states to be finite. At this point, the reader might have the feeling that these results are not really about cellular automata, and there are at least three answers to that. The first answer is that one could easily impose the shift and simultaneously prevent the use of a highly redundant "fully shifted local transition function", but this requires using a category of configurations instead of a poset of configurations. The former is very similar to the poset, except that the yes/no question "is this configuration a subconfiguration of this other one ?" is replaced by the open-ended question "where does this configuration appear in this other one ?" [5]. The goal of this paper is indeed to introduce the concepts needed for this other point of view, among many others. The second answer is that the proofs are more about the decomposition/composition process involved in the local/global definition of cellular automata. Because of the simplicity of cellular spaces, groups, the description is very simple to make "directly". In other situations, a Kan extension presentation is the most effective way to describe the spatial extensions/restriction, for example when the space is an evolving graph [2, 5]. The third answer is that, with small modifications, this result is closely related to the Curtis-Hedlund theorem. Indeed, if one restores the finiteness of the set of states constraints, one can see that the poset of finite support configurations is a "generating" part of the poset of open subsets of the product topology. In this case, the fine transition function $\phi$ can be viewed as encoding an important part of the topological behavior of the global transition function $\Delta$ [1, 3].

To finish, let us mention an important aspect of the Kan extensions considered here and in other papers of the authors [2, 5]. They have the property to be *pointwise*. Intuitively, this means that they can be computed "algorithmically" using simple building blocks. This formulation in terms of building blocks is completely equivalent and is the one used in the other papers, firstly because it is via these building blocks that the authors discovered these links between spatially-extended dynamical systems and category theory, and secondly because this formulation is closer to the software implementations of the considered models. In fact, it is possible to have an implementation completely generic over the particular kind of space considered, *e.g.* evolving graphs of any sort, evolving higher-order structures such as abstract cell [5], evolving strings such as Lindenmayer systems [2], or Cayley graphs as considered here.

## References


1 Tullio Ceccherini-Silberstein and Michel Coornaert. *Cellular automata and groups*. Springer Science & Business Media, 2010.

2 Alexandre Fernandez, Luidnel Maignan, and Antoine Spicher. Lindenmayer systems and global transformations. In Ian McQuillan and Shinnosuke Seki, editors, *Unconventional Computation and Natural Computation - 18th International Conference, UCNC 2019, Tokyo, Japan, June 3-7, 2019, Proceedings*, volume 11493 of *Lecture Notes in Computer Science*, pages 65–78. Springer, 2019. doi:10.1007/978-3-030-19311-9_7.

3 Gustav A Hedlund. Endomorphisms and automorphisms of the shift dynamical system. *Mathematical systems theory*, 3(4):320–375, 1969.

4 S. MacLane. *Categories for the Working Mathematician*. Graduate Texts in Mathematics. Springer New York, 2013.

5 Luidnel Maignan and Antoine Spicher. Global graph transformations. In Detlef Plump, editor, *Proceedings of the 6th International Workshop on Graph Computation Models co-located with the 8th International Conference on Graph Transformation (ICGT 2015) part of the Software Technologies: Applications and Foundations (STAF 2015) federation of conferences, L'Aquila, Italy, July 20, 2015*, volume 1403 of *CEUR Workshop Proceedings*, pages 34–49. CEUR-WS.org, 2015. URL: http://ceur-ws.org/Vol-1403/paper4.pdf.

# Conjunctive Grammars, Cellular Automata and Logic

**Théo Grente**
GREYC, Université de Caen Normandie, France

**Étienne Grandjean**
GREYC, Université de Caen Normandie, France

## Abstract

The expressive power of the class `Conj` of *conjunctive languages*, i.e. languages generated by the *conjunctive grammars* of Okhotin, is largely unknown, while its restriction `LinConj` to *linear conjunctive grammars* equals the class of languages recognized by *real-time one-dimensional one-way* cellular automata. We prove two weakened versions of the open question `Conj` $\subseteq ?$ `RealTime1CA`, where `RealTime1CA` is the class of languages recognized by *real-time one-dimensional two-way* cellular automata:

1. it is true for *unary* languages;
2. `Conj` $\subseteq$ `RealTime2OCA`, i.e. any conjunctive language is recognized by a *real-time two-dimensional one-way* cellular automaton.

Interestingly, we express the rules of a conjunctive grammar in two *Horn logics*, which *exactly characterize* the complexity classes `RealTime1CA` and `RealTime2OCA`.

**2012 ACM Subject Classification** Theory of computation → Complexity theory and logic; Theory of computation → Formal languages and automata theory

**Keywords and phrases** Computational complexity, Real-time, One-dimensional/two-dimensional cellular automaton, One-way/two-way communication, Grid-circuit, Unary language, Descriptive complexity, Existential second-order logic, Horn formula

**Digital Object Identifier** 10.4230/OASIcs.AUTOMATA.2021.8

**Funding** This work has been partly supported by the PING/ACK project of the French National Agency for Research (ANR-18-CE40-0011).

**Acknowledgements** This paper would not exist without the inspiration of Véronique Terrier. Her in-depth knowledge of cellular automata, their complexity classes and their closure properties in relation to formal language theory, the references and advice she generously gave us, as well as her careful reading, were essential in designing and finalizing the results and the presentation of the paper. E.g., the class diagram of Figure 8 is due to her. We also thank the three reviewers of this paper for their careful reading and their detailed constructive comments and suggestions which helped us improve some parts of this paper.

## 1 Introduction

For decades, logic has maintained close relationships with, on the one hand, computational models [31] and computational complexity [3], in particular through descriptive complexity [7, 16, 21, 11, 14, 2], and on the other hand with formal language theory and grammars [8, 21].

**Conjunctive grammars versus logic.** Okhotin [26] wrote that "context-free grammars may be thought of as a *logic* for inductive description of syntax in which the propositional connectives available... are restricted to *disjunction only*". Thus, twenty years ago, the same author introduced *conjunctive grammars* [22] as an extension of context-free grammars by adding an explicit *conjunction* operation within the grammar rules.


As shown by Okhotin [22], conjunctive grammars – and more generally, Boolean grammars [24, 26] – inherit the parsing algorithms of the ordinary context-free grammars, without increasing their computational complexity. However, the expressive power of these grammars is largely unknown. The fact that the class Conj of languages generated by conjunctive grammars has many closure properties – it is trivially closed under reverse, concatenation, Kleene closure, disjunction and conjunction – suggests that this class has equivalent definitions in computational complexity and/or logic.

**Conjunctive grammars versus real-time cellular automata.** Note that the LinConj subclass of languages generated by *linear* conjunctive grammars was found to be equal to the Trellis class of languages recognized by *trellis* automata [25], or equivalently, *one-way real-time* cellular automata. Faced with this result, it is tempting to ask the following question: is the larger class Conj equal to the class RealTime1CA of languages recognized by *two-way* real-time cellular automata? Either answer to this question has strong consequences:

- If $\texttt{Conj} = \texttt{RealTime1CA}$ then each of the two classes will benefit from the closure properties of the other class; in particular, RealTime1CA would be closed under reverse, which was shown by [15] to imply $\texttt{RealTime1CA} = \texttt{LinearTime}_{\texttt{1CA}}$, i.e. real-time is nothing but *linear time* for cellular automata, a surprising positive answer to a longstanding open question [6, 28, 30].
- If $\texttt{Conj} \neq \texttt{RealTime1CA}$ then $\texttt{Conj} \subsetneq \texttt{DSPACE}(n)$ or $\texttt{RealTime1CA} \subsetneq \texttt{DSPACE}(n)$: any of these strict inclusions would be a striking result.

Real-time is the minimal time of cellular automata (CA). Recall that RealTime1CA (resp. Trellis) is the class of languages recognized in real-time by *one-dimensional* CA with *two-way* (resp. *one-way*) communication and input word given in parallel. We know the strict inclusion $\texttt{Trellis} \subsetneq \texttt{RealTime1CA}$. The robustness of these classes is attested by their characterization by two sub-logics of ESO – the *existential second-order* logic, which characterizes NP – with *Horn formulas* as their first-order parts[1], and called respectively pred-ESO-HORN and incl-ESO-HORN, see [12, 13]. For short, we write $\texttt{RealTime1CA} = \texttt{pred-ESO-HORN}$ and $\texttt{Trellis} = \texttt{incl-ESO-HORN}$.

**Results of this paper.** This paper focuses on the relationships between the class of conjunctive languages and the real-time classes of cellular automata. Although we do not know the answer to the question $\texttt{Conj} =? \texttt{RealTime1CA}$ or even to the question of the inclusion $\texttt{Conj} \subseteq? \texttt{RealTime1CA}$, we prove two weakened versions of this inclusion:

1. $\texttt{Conj}_1 \subseteq \texttt{RealTime1CA}_1$: The inclusion holds when restricted to *unary* languages[2].
2. $\texttt{Conj} \subseteq \texttt{RealTime2OCA}$: The inclusion holds for real-time of *two-dimensional one-way* cellular automata (2-OCA). (We have $\texttt{RealTime1CA} \subseteq \texttt{RealTime2OCA}$.)

To grasp the scope of inclusion (1), it is important to note that unlike the subclass $\texttt{CFL}_1$ of the unary languages of the class of context-free languages, which is reduced to regular languages, $\texttt{CFL}_1 = \texttt{Reg}_1$, the class $\texttt{Conj}_1$ was shown by Jez [17] to be much larger than $\texttt{Reg}_1$. Understanding its precise expressiveness seems as difficult a problem to us as for Conj.

Our inclusion (2) improves the inclusion $\texttt{CFL} \subseteq \texttt{RealTime2SOCA}$, where RealTime2SOCA denotes the class of languages recognized by real-time *sequential* two-dimensional one-way cellular automata, proved by Terrier [29], who uses a result by King [18] and improves results by Kosaraju [20] and Chang et al. [4]. Terrier's result derives transitively from (2): $\texttt{CFL} \subseteq \texttt{Conj} \subseteq \texttt{RealTime2OCA} \subseteq \texttt{RealTime2SOCA}$.

---

[1] The class ESO-HORN of languages defined by existential second-order formulas with Horn formulas as their first-order parts is exactly PTIME, see [10, 11].

[2] The subclass of the unary languages of a class of languages $\mathcal{C}$ is denoted $\mathcal{C}_1$.

Inclusion (2) seems difficult to improve. Since any problem in `RealTime1CA` is decidable in time $O(n^2)$ (by a RAM algorithm), the hypothetical inclusion `Conj` $\subseteq$ `RealTime1CA` implies that any conjunctive language is decidable in time $O(n^2)$: this would be a breakthrough!

**Logic as a bridge from problems and grammars to real-time CAs.** Logic has been the basis of logic programming and database queries for decades, especially Horn logic through the Prolog and Datalog programming languages [1, 19, 11]. Likewise, the above-mentioned logical characterizations of real-time complexity classes of CAs, `RealTime1CA` = `pred-ESO-HORN` and `Trellis` = `incl-ESO-HORN`, have been used to easily show that several problems belong to the `RealTime1CA` or `Trellis` class by inductively expressing/programming the problems in the corresponding Horn logic, see [12, 13].

In this paper, the same logic programming method is adopted. We prove inclusion (1) $\texttt{Conj}_1 \subseteq \texttt{RealTime1CA}_1$ by expressing a unary language generated by a conjunctive grammar in the `pred-ESO-HORN` logic. Inclusion (1) follows, by the equality `pred-ESO-HORN` = `RealTime1CA`. Similarly, to prove inclusion (2) `Conj` $\subseteq$ `RealTime2OCA`, we first design a logic denoted `incl-pred-ESO-HORN` so that `incl-pred-ESO-HORN` = `RealTime2OCA`. Then, we express any conjunctive language in this logic, proving that it belongs to `RealTime2OCA`, as claimed. Thus, the heart of each proof consists in presenting a formula of a certain *Horn logic*, which *inductively* expresses how a word is generated by a conjunctive grammar: the Horn clauses of the formula *naturally* imitate the rules of the grammar.

**Our proof method and the paper structure.** After Section 2 gives some definitions, Sections 3 and 4 present inclusions (1) and (2) and their proofs with a common plan: Subsection 3.1 (resp. 4.1) expresses the inductive generating process of a conjunctive grammar, assumed in binary (Chomsky) normal form in the logic `pred-ESO-HORN` (resp. `incl-pred-ESO-HORN`). Subsection 3.2 (resp. 4.2) shows that any formula of this logic can be normalized into a formula which mimics the computation of a two-dimensional (resp. three-dimensional) *grid-circuit* called `Grid` (resp. `Cube`); Subsection 3.3 (resp. 4.3) translates the grid-circuit into a real-time one-dimensional CA (resp. two-dimensional OCA). Note that we prove the equivalence of our logics with grid-circuits and CA real-time[3]: `pred-ESO-HORN` = `Grid` = `RealTime1CA` and `incl-pred-ESO-HORN` = `Cube` = `RealTime2OCA`. Section 5 deals briefly with the meaning of our results and open problems around a diagram of the known relations between the `Conj` class and the CA complexity classes studied here, for the general case and for the unary case.

# 2 Preliminaries

## 2.1 Conjunctive grammars and their binary normal form

Conjunctive grammars extend context-free grammars with a conjunction operation.

▶ **Definition 1** (Conjunctive grammar, conjunctive language). *[22, 23]*

*A* conjunctive grammar *is a tuple* $G = (\Sigma, N, P, S)$ *where* $\Sigma$ *is the finite set of* terminal *symbols,* $N$ *is the finite set of* nonterminal *symbols,* $S \in N$ *is the* initial *symbol, and* $P$ *is the finite set of* rules*, each of the form* $A \to \alpha_1 \& ... \& \alpha_m$*, for* $m \geq 1$ *and* $\alpha_i \in (\Sigma \cup N)^+$*.*

[3] We have chosen to give here a simplified proof of the logical characterization `pred-ESO-HORN` = `Grid` = `RealTime1CA` already proved in [12] so that this paper is self-content, but above all because our proof of the similar result `incl-pred-ESO-HORN` = `Cube` = `RealTime2OCA` is an extension of it.

- *The set of words* $L(A) \subseteq \Sigma^+$ generated *by any* $A \in N$ *is defined by induction: if the rules for* $A$ *are* $A \to \alpha_1^1 \& ... \& \alpha_{m_1}^1 \mid \cdots \mid \alpha_1^k \& ... \& \alpha_{m_k}^k$*, then* $L(A) \coloneqq \bigcup_{i=1}^{k} \bigcap_{j=1}^{m_i} L(\alpha_j^i)$*. (As usual, take the least solution of the language equations defining the sets* $L(A)$*, for* $A \in N$*.)*
- *The* language generated *by the grammar* $G$ *is* $L(S)$*. It is called a* conjunctive language*.*

Okhotin [26] gave many examples of conjunctive languages which are not context-free. Moreover, Jez [17] proved that there are such languages on unary alphabet, in particular, the set $\{a^{4^k} \mid k \in \mathbb{N}\}$ is a conjunctive language which is not context-free (= not regular).

We will mainly use the binary normal form of conjunctive grammars, which extends the Chomsky normal form of context-free grammars. Each conjunctive grammar can be rewritten in an equivalent binary normal form [22, 26].

▶ **Definition 2** (Binary normal form [22])**.** *A conjunctive grammar* $G = (\Sigma, N, P, S)$ *is in* binary normal form *if each rule in* $P$ *has one of the two following forms:*
- *a long rule:* $A \to B_1C_1 \& ... \& B_mC_m$ $(m \geq 1, B_i, C_j \in N)$*;*
- *a short rule:* $A \to a$ $(a \in \Sigma)$*.*

## 2.2 Elements of logic

The underlying structure we will adopt to encode an input word $w = w_1 \dots w_n$ over its index interval $[1, n] = \{1, \dots, n\}$ uses the *successor* and *predecessor* functions and the monadic predicates `min` and `max` as its *only* arithmetic functions/predicates:

▶ **Definition 3** (structure encoding a word)**.** *Each nonempty word* $w = w_1 \dots w_n \in \Sigma^n$ *on a fixed finite alphabet* $\Sigma$ *is represented by the first-order structure*
$\langle w \rangle \coloneqq ([1, n]; (Q_s)_{s \in \Sigma}, \texttt{min}, \texttt{max}, \texttt{suc}, \texttt{pred})$
*of domain* $[1, n]$*, monadic predicates* $Q_s$*,* $s \in \Sigma$*,* `min` *and* `max` *such that* $Q_s(i) \iff w_i = s$*,* $\texttt{min}(i) \iff i = 1$*, and* $\texttt{max}(i) \iff i = n$*, and unary functions* `suc` *and* `pred` *such that* $\texttt{suc}(i) = i + 1$ *for* $i < n$ *and* $\texttt{suc}(n) = n$*,* $\texttt{pred}(i) = i - 1$ *for* $i > 1$ *and* $\texttt{pred}(1) = 1$*.*
*Let* $\mathcal{S}_\Sigma$ *denote the signature* $\{(Q_s)_{s \in \Sigma}, \texttt{min}, \texttt{max}, \texttt{suc}, \texttt{pred}\}$ *of the structure* $\langle w \rangle$*.*

▶ **Notation 1.** *Let* $x + k$ *and* $x - k$ *abbreviate the terms* $\texttt{suc}^k(x)$ *and* $\texttt{pred}^k(x)$*, for a fixed integer* $k \geq 0$*. We will also use the intuitive abbreviations* $x = 1$*,* $x = n$ *and* $x > k$*, for a fixed integer* $k \geq 1$*, in place of the formulas* $\texttt{min}(x)$*,* $\texttt{max}(x)$ *and* $\neg\texttt{min}(x - (k - 1))$*, respectively.*

## 2.3 Cellular automata and real-time

▶ **Definition 4** (1-CA and 2-0CA)**.** *A* $d$-dimensional cellular automaton *(CA) is a triple* $(\texttt{S}, \mathcal{N}, \texttt{f})$ *where* `S` *is the finite set of* states*,* $\mathcal{N} \subset \mathbb{Z}^d$ *is the* neighborhood*, and* $\texttt{f} : \texttt{S}^{|\mathcal{N}|} \to \texttt{S}$ *is the* transition function*. We are interested in the following two special cases:*
- 1-CA*: It is a* one-dimensional two-way *cellular automaton* $(\texttt{S}, \{-1, 0, 1\}, \texttt{f})$*, for which the state* $\langle c, t \rangle$ *of any cell* $c$ *at a time* $t > 1$ *is updated in this way:*
  $\langle c, t \rangle = \texttt{f}(\langle c - 1, t - 1 \rangle, \langle c, t - 1 \rangle, \langle c + 1, t - 1 \rangle)$*.*
- 2-OCA*: It is a* two-dimensional one-way *cellular automaton* $(\texttt{S}, \{(0, 0), (-1, 0), (0, -1)\}, \texttt{f})$ *for which the state* $\langle c_1, c_2, t \rangle$ *of any cell* $(c_1, c_2)$ *at a time* $t > 1$ *is updated in this way:*
  $\langle c_1, c_2, t \rangle = \texttt{f}(\langle c_1, c_2, t - 1 \rangle, \langle c_1 - 1, c_2, t - 1 \rangle, \langle c_1, c_2 - 1, t - 1 \rangle)$*.*

▶ **Definition 5** (permanent and quiescent states)**.** *In a CA, a state* $\sharp$ *is* permanent *if a cell in state* $\sharp$ *remains in this state forever. A state* $\lambda$ *of a CA is* quiescent *if a cell in state* $\lambda$ *remains in this state as long as the states of its neighborhood cells are quiescent or permanent.*

▶ **Definition 6** (CA as a word acceptor). *A cellular automaton* $(\mathtt{S}, \mathcal{N}, \mathtt{f})$ *with an* input alphabet $\Sigma \subset \mathtt{S}$, *a* permanent state $\sharp$, *a* quiescent state $\lambda$, *and a set of* accepting states $\mathtt{S}_{\mathtt{acc}} \subset \mathtt{S}$ *acts as a* word acceptor *if it operates on an input word* $w \in \Sigma^+$ *in respecting the following conditions (see Figure 1).*

***Input.*** *For a* 1*-CA, the* $i$*-th symbol of the input* $w = w_1 \dots w_n$ *is given to the cell* $i$ *at the initial time* 1*:* $\langle i, 1\rangle = w_i$. *All other cells are in the permanent state* $\sharp$. *For a* 2*-OCA, the* $i$*-th symbol of the input is given to the cell* $(i, 1)$ *at time* 1*:* $\langle i, 1, 1\rangle = w_i$. *At time* 1*, the cells* $(c_1, c_2) \in [1, n] \times [2, n]$ *are in the quiescent state* $\lambda$*, all other cells are in the permanent state* $\sharp$.

***Output.*** *One specific cell called the* output cell *gives the output, "accept" or "reject", of the computation. For a* 1*-CA, the output cell is the cell* 1. *For a* 2*-OCA, the output cell is* $(n, n)$.

***Acceptance.*** *An input word is* accepted *by a* 1*-CA (resp.* 2*-CA)* at time $t$ *if the output cell enters an accepting state at time* $t$.

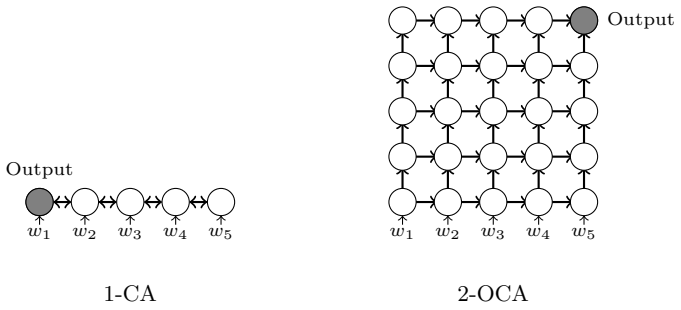



**Figure 1** Input and output of a CA acting as a word acceptor.

▶ **Definition 7** (`RealTime1CA`, `RealTime2OCA`). *A word is* accepted in real-time *by a* 1*-CA (resp.* 2*-OCA) if the word is accepted in minimal time for the output cell* 1 *(resp.* $(n, n)$*) to receive each of its letters. A language is* recognized in real-time *by a CA if it is the set of words that it accepts in real-time. The class* `RealTime1CA` *(resp.* `RealTime2OCA`*) is the class of languages recognized in real-time by a* 1*-CA (resp.* 2*-OCA).*

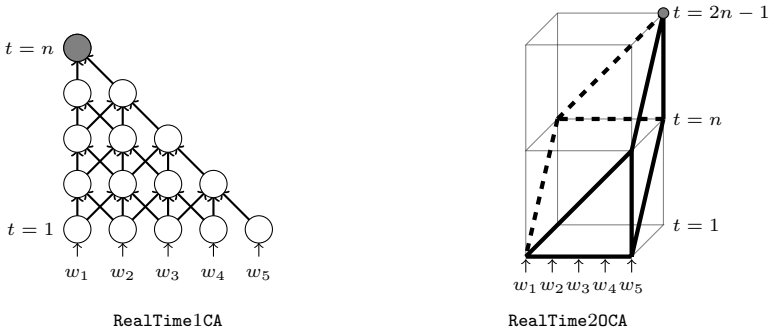



**Figure 2** Space-time diagrams of `RealTime1CA` and `RealTime2OCA`.

## 3 Real-time recognition of a unary conjunctive language

In this section, we prove our first main result:

▶ **Theorem 8.** $\mathtt{Conj}_1 \subseteq \mathtt{RealTime1CA}_1$.

## 3.1 Expressing inductively a unary conjunctive language in logic

The generating process of a unary conjunctive language is naturally expressed in the logic pred-ESO-HORN, an inductive Horn logic whose only function is the predecessor function.

▶ **Definition 9** (pred-ESO-HORN). *A formula of* pred-ESO-HORN *is a formula* $\Phi := \exists \mathbf{R} \forall x \forall y \psi(x,y)$ *where* $\mathbf{R}$ *is a finite set of binary predicates and* $\psi$ *is a conjunction of Horn clauses, of signature* $\mathcal{S}_\Sigma \cup \mathbf{R}$*, and of one the three following forms:*

- an input clause: $\mathtt{min}(x) \wedge (\neg)\,\mathtt{min}(y) \wedge Q_s(y) \rightarrow R(x,y)$ *with* $s \in \Sigma$ *and* $R \in \mathbf{R}$*;*
- a computation clause: $\delta_1 \wedge \ldots \wedge \delta_r \rightarrow R(x,y)$ *with* $R \in \mathbf{R}$ *and where each hypothesis* $\delta_h$ *is an atom* $S(x,y)$ *or a conjunction* $S(x-i, y-j) \wedge x > i \wedge y > j$*, with* $S \in \mathbf{R}$ *and* $i, j \geq 0$ *two integers such that* $i + j > 0$*;*
- a contradiction clause: $\mathtt{max}(x) \wedge \mathtt{max}(y) \wedge R(x,y) \rightarrow \bot$ *with* $R \in \mathbf{R}$.

By abuse of notation, let us also call pred-ESO-HORN the class of languages defined by a formula of pred-ESO-HORN.

▶ **Notation 2.** *We will freely use equalities (resp. inequalities)* $x = i$ *and* $y = j$ *(resp.* $x > i$*,* $y > j$*), for constants* $i, j$*, in our formulas since they can be easily defined in* pred-ESO-HORN*. For example, the binary predicate* $R^{x>2}$ *of intuitive meaning* $R^{x>2}(x,y) \iff x > 2$ *is defined inductively by the following clauses where* $R^{x=a}(x,y)$ *means* $x = a$*:*

- $\mathtt{min}(x) \rightarrow R^{x=1}(x,y)$*;* $x > 1 \wedge R^{x=1}(x-1,y) \rightarrow R^{x=2}(x,y)$*;*
- $x > 1 \wedge R^{x=2}(x-1,y) \rightarrow R^{x>2}(x,y)$*;* $x > 1 \wedge R^{x>2}(x-1,y) \rightarrow R^{x>2}(x,y)$.

*Also, some other arithmetic predicates easily defined in* pred-ESO-HORN *will be used. For example,* $y = 2x$ *can be replaced by the atom* $R^{y=2x}(x,y)$*, where* $R^{y=2x}$ *is defined by the following two clauses using the predicates* $R^{x=1}, R^{y=2}, R^{x>1}$ *and* $R^{y>2}$*:*

- $x = 1 \wedge y = 2 \rightarrow R^{y=2x}(x,y)$ *;* $x > 1 \wedge y > 2 \wedge R^{y=2x}(x-1, y-2) \rightarrow R^{y=2x}(x,y)$.

▶ **Notation 3.** *More generally, let* $R^{\rho(x,y)}$ *denote a binary predicate whose meaning is* $R^{\rho(x,y)}(x,y) \iff \rho(x,y)$*, for a property or a formula* $\rho(x,y)$*. We will also use a set of binary arithmetic predicates denoted by* $\mathbf{R}_{\mathtt{arith}}$*, which consists of* $R^{x=y}, R^{y=2x}$ *and* $R^{\rho(x,y)}$*, for* $\rho(x,y) := x \geq \lceil \frac{y}{2} \rceil$*, and the predicates used to define them in* pred-ESO-HORN.

Let us prove that for every unary conjunctive language, its complement can be defined in $\mathtt{pred\text{-}ESO\text{-}HORN}_1$.

▶ **Lemma 10.** *For each language* $L \subseteq a^+$*, if* $L \in \mathtt{Conj}_1$ *then* $a^+ \setminus L \in$ pred-ESO-HORN.

**Proof.** Let $G = (\{a\}, N, P, S)$ be a conjunctive grammar in binary normal form which generates $L$. For each $A \in N$ and each unary word $a^y$, we have, according to the length $y$, the following equivalences which will be the basis of our induction:

- if $y = 1$, then $a^y = a \in L(A) \iff$ the short rule $A \rightarrow a$ belongs to $P$;
- if $y > 1$, then $a^y \in L(A) \iff$ there is a long rule $A \rightarrow B_1C_1 \& \ldots \& B_mC_m$ in $P$ such that, for each $i \in \{1, \ldots, m\}$, there exists $x \geq \lceil \frac{y}{2} \rceil$ such that either $a^x \in L(B_i)$ and $a^{y-x} \in L(C_i)$, or $a^{y-x} \in L(B_i)$ and $a^x \in L(C_i)$.

We want to construct a first-order formula $\forall x \forall y \psi_G(x,y)$ of signature $\mathcal{S}_\Sigma \cup \mathbf{R}$, for $\Sigma := \{a\}$ and the set of binary predicates $\mathbf{R} := \{\mathtt{Maj}_A, \mathtt{Min}_A \mid A \in N\} \cup \{\mathtt{Sum}_{BC} \mid B, C \in N\} \cup \mathbf{R}_{\mathtt{arith}}$ so that the formula $\Phi_G := \exists \mathbf{R} \forall x \forall y \psi_G$ belongs to pred-ESO-HORN and defines the language $a^+ \setminus L$. The intuitive meanings of the predicates $\mathtt{Maj}_A$, $\mathtt{Min}_A$ and $\mathtt{Sum}_{BC}$ are as follows:

- $\mathtt{Maj}_A(x,y) \iff \lceil \frac{y}{2} \rceil \leq x \leq y$ and $a^x \in L(A)$;
- $\mathtt{Min}_A(x,y) \iff \lceil \frac{y}{2} \rceil \leq x < y$ and $a^{y-x} \in L(A)$ ;

- $\texttt{Sum}_{\texttt{BC}}(x,y) \iff$ there is some $x'$ with $\lceil \frac{y}{2} \rceil \leq x' \leq x$ such that either $a^{x'} \in L(B)$ and $a^{y-x'} \in L(C)$, or $a^{y-x'} \in L(B)$ and $a^{x'} \in L(C)$.

Note that for $x = y$, the above equivalence for $\texttt{Maj}_A$ implies $\texttt{Maj}_A(x,y) \iff a^y \in L(A)$.

Let us give and justify a list of Horn clauses whose conjunction $\psi'_G$ defines the predicates $\texttt{Maj}_A$, $\texttt{Min}_A$ and $\texttt{Sum}_{BC}$, using the arithmetic predicates of $\mathbf{R}_{\texttt{arith}}$ (see Notations 2 and 3), namely $R^{x=y}, R^{y=2x}$ and $R^{\rho(x,y)}$, for $\rho(x,y) \coloneqq x \geq \lceil \frac{y}{2} \rceil$.

**Short rules.** Each rule $A \to a$ of $P$ is expressed by the input clause:

- $\texttt{min}(x) \wedge \texttt{min}(y) \wedge Q_a(y) \to \texttt{Maj}_A(x,y)$.

**Induction on the length $y$.** If we have for $y > 1$ the inequalities $\lceil \frac{y-1}{2} \rceil \leq x \leq y-1$ and $x \geq \lceil \frac{y}{2} \rceil$ then $\lceil \frac{y}{2} \rceil \leq x \leq y$. This justifies the clause:

- $y > 1 \wedge \texttt{Maj}_A(x,y-1) \wedge x \geq \lceil \frac{y}{2} \rceil \to \texttt{Maj}_A(x,y)$ for all $A \in N$.

For $y > 1$ and $y = 2x$, we have $a^x = a^{y-x}$ and $\lceil \frac{y}{2} \rceil \leq x < y$. This justifies the clause:

- $y > 1 \wedge \texttt{Maj}_A(x,y-1) \wedge y = 2x \to \texttt{Min}_A(x,y)$ for all $A \in N$.

If for $x, y > 1$ we have the inequalities $\lceil \frac{y-1}{2} \rceil \leq x-1 < y-1$, then $\lceil \frac{y}{2} \rceil \leq x < y$. Moreover, $a^{(y-1)-(x-1)} = a^{y-x}$. This justifies the clause:

- $x > 1 \wedge y > 1 \wedge \texttt{Min}_A(x-1,y-1) \to \texttt{Min}_A(x,y)$ for all $A \in N$.

**Concatenation.** For all $B, C \in N$, it is clear that the concatenation predicate $\texttt{Sum}_{\texttt{BC}}$ is defined inductively by the following three clauses:

- *initialization:* $\texttt{Maj}_B(x,y) \wedge \texttt{Min}_C(x,y) \to \texttt{Sum}_{\texttt{BC}}(x,y)$ ;
  $\texttt{Min}_B(x,y) \wedge \texttt{Maj}_C(x,y) \to \texttt{Sum}_{\texttt{BC}}(x,y)$;
- *induction:* $\neg\texttt{min}(x) \wedge \texttt{Sum}_{\texttt{BC}}(x-1,y) \to \texttt{Sum}_{\texttt{BC}}(x,y)$.

**Long rules.** Each rule $A \to B_1C_1\& \ldots \& B_mC_m$ of $P$ is expressed by the clause:

- $x = y \wedge \texttt{Sum}_{\texttt{B}_1\texttt{C}_1}(x,y) \wedge \cdots \wedge \texttt{Sum}_{\texttt{B}_m\texttt{C}_m}(x,y) \to \texttt{Maj}_A(x,y)$.

Thus, the formula $\forall x \forall y \psi'_G$ where $\psi'_G$ is the conjunction of the above clauses defines the predicates $\texttt{Maj}_A, \texttt{Min}_A$, and $\texttt{Sum}_{BC}$.

**Definition of $a^+ \setminus L$.** We have the equivalence $\texttt{Maj}_S(n,n) \iff a^n \in L(S) \iff a^n \in L$. Therefore, the following contradiction clause expresses $a^n \notin L$:

- $\gamma_S \coloneqq \texttt{max}(x) \wedge \texttt{max}(y) \wedge \texttt{Maj}_S(x,y) \to \bot$.

Finally, observe that the formula $\Phi_G \coloneqq \exists \mathbf{R} \forall x \forall y \psi_G$ where $\psi_G$ is $\gamma_{\texttt{arith}} \wedge \psi'_G \wedge \gamma_S$ and $\gamma_{\texttt{arith}}$ is the conjunction of clauses that defines the arithmetic predicates of $\mathbf{R}_{\texttt{arith}}$, belongs to pred-ESO-HORN. Since we have $\langle a^n \rangle \models \Phi_G \iff a^n \notin L$, as justified above, then the language $a^+ \setminus L$ belongs to pred-ESO-HORN, as claimed. ◀

## 3.2 Equivalence of logic with grid-circuits

We introduce the *grid-circuit* as an intermediate object between our logic and the real-time cellular automaton: see Figure 3.

▶ **Definition 11.** *A* grid-circuit *is a tuple* $\mathcal{C} \coloneqq (\Sigma, (\texttt{Input}_n)_{n>0}, \mathbf{Q}, \mathbf{Q}_{\texttt{acc}}, \texttt{g})$ *where*

- $\Sigma$ *is the* input alphabet *and* $(\texttt{Input}_n)_{n>0}$ *is the family of* input functions $\texttt{Input}_n : \Sigma^n \times [1,n]^2 \to \Sigma \cup \{\$\}$ *such that, for* $w = w_1 \ldots w_n \in \Sigma^n$, $\texttt{Input}_n(w,x,y) = w_y$ *if* $x = 1$ *and* $\texttt{Input}_n(w,x,y) = \$$ *otherwise,*
- $\mathbf{Q} \cup \{\sharp\}$ *is the finite set of* states *and* $\mathbf{Q}_{\texttt{acc}} \subseteq \mathbf{Q}$ *is the subset of* accepting states,
- $\texttt{g} : (\mathbf{Q} \cup \{\sharp\})^2 \times (\Sigma \cup \{\$\}) \to \mathbf{Q}$ *is the* transition function.

▶ **Definition 12** (computation of a grid-circuit). *The* computation $\mathcal{C}_w$ *of a grid-circuit* $\mathcal{C} := (\Sigma, (\mathtt{Input}_n)_{n>0}, \mathbf{Q}, \mathbf{Q}_{\mathtt{acc}}, \mathtt{g})$ *on a* $w = w_1 \dots w_n \in \Sigma^n$ *is a regular grid of* $(n+1)^2$ *sites* $(x, y) \in [0, n]^2$*, each in a state* $\langle x, y\rangle \in \mathbf{Q} \cup \{\sharp\}$ *computed inductively:*

- *each site in* $\{0\} \times [0, n]$ *or* $[0, n] \times \{0\}$ *is in the particular state* $\sharp$*;*
- *the state of each site* $(x, y) \in [1, n]^2$ *is* $\langle x, y\rangle = \mathtt{g}(\langle x, y-1\rangle, \langle x-1, y\rangle, \mathtt{Input}_n(w, x, y))$.

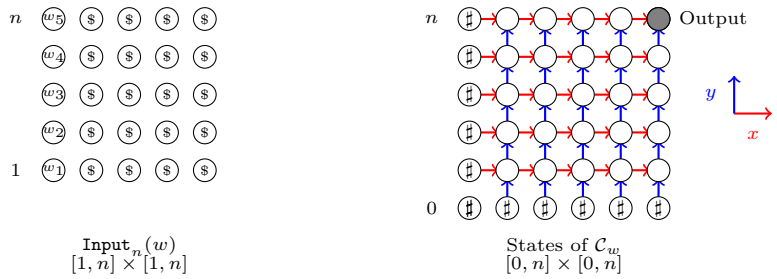



**Figure 3** The grid-circuit.

A word $w = w_1 \dots w_n \in \Sigma^n$ is *accepted* by the grid-circuit $\mathcal{C}$ if the output state $\langle n, n\rangle$ of $\mathcal{C}_w$ belongs to $\mathbf{Q}_{\mathtt{acc}}$. The language *recognized* by $\mathcal{C}$ is the set of words it accepts. We denote by `Grid` the class of languages recognized by a grid-circuit.

Actually, our predecessor Horn logic is equivalent to grid-circuits.

▶ **Lemma 13** ([12]). `pred-ESO-HORN` = `Grid`.

**Proof.** In some sense, a grid-circuit is the "normalized form" of a formula of `pred-ESO-HORN`. So, the inclusion `Grid` $\subseteq$ `pred-ESO-HORN` is proved straightforwardly.

The first step of the proof of the converse inclusion `pred-ESO-HORN` $\subseteq$ `Grid` is to show that every formula $\Phi := \exists \mathbf{R} \forall x \forall y \psi(x, y)$ in `pred-ESO-HORN` is equivalent to a formula $\Phi' \in$ `pred-ESO-HORN` in which the only hypotheses of computation clauses are atoms $S(x, y)$ and conjunctions $S(x-1, y) \wedge x > 1$ and $S(x, y-1) \wedge y > 1$ .

**Elimination of atoms $R(x-i, y-j)$ for $i+j > 1$.** The idea is to introduce new "shift" predicates $R^{x-i', y-j'}$ for fixed integers $i', j' > 0$ with the intuitive meaning:
$R^{x-i', y-j'}(x, y) \iff R(x-i', y-j') \wedge x > i' \wedge y > j'$.
Let us explain the method by an example. Assume we have in $\psi$ the Horn clause
(1) $x > 3 \wedge y > 2 \wedge S(x-3, y-2) \to T(x, y)$. This clause is replaced by the clause
(2) $S^{x-2, y-2}(x-1, y) \wedge x > 1 \to T(x, y)$
for which the predicates $S^{x-1}$, $S^{x-2}$, $S^{x-2, y-1}$ and $S^{x-2, y-2}$ are defined by the respective clauses: $x > 1 \wedge S(x-1, y) \to S^{x-1}(x, y)$, $x > 1 \wedge S^{x-1}(x-1, y) \to S^{x-2}(x, y)$, $y > 1 \wedge S^{x-2}(x, y-1) \to S^{x-2, y-1}(x, y)$, and $y > 1 \wedge S^{x-2, y-1}(x, y-1) \to S^{x-2, y-2}(x, y)$, which imply together the clause $x > 2 \wedge y > 2 \wedge S(x-2, y-2) \to S^{x-2, y-2}(x, y)$ and then also $x > 3 \wedge y > 2 \wedge S(x-3, y-2) \to S^{x-2, y-2}(x-1, y)$.

It is clear that the formula $\Phi := \exists \mathbf{R} \forall x \forall y \psi$ is equivalent to the formula $\Phi' := \exists \mathbf{R}' \forall x \forall y \psi'$ where $\mathbf{R}' := \mathbf{R} \cup \{S^{x-1}, S^{x-2}, S^{x-2, y-1}, S^{x-2, y-2}\}$ and $\psi'$ is the conjunction $\psi_{\mathtt{replace}} \wedge \psi_{\mathtt{def}}$, where $\psi_{\mathtt{replace}}$ is the formula $\psi$ in which clause (1) is replaced by clause (2), and $\psi_{\mathtt{def}}$ is the conjunction of the above clauses defining the new predicates of $\mathbf{R}'$.

Thus, any formula $\Phi \in$ `pred-ESO-HORN` is equivalent to a formula $\Phi' \in$ `pred-ESO-HORN` whose computation clauses only contain hypotheses of the following three forms:
$R(x-1, y) \wedge x > 1$ ; $R(x, y-1) \wedge y > 1$ ; $R(x, y)$. The next step is to eliminate these $R(x, y)$.

**Elimination of hypotheses $R(x,y)$.** (sketch of proof): The first idea is to group together in each computation clause the hypothesis atoms of the form $R(x,y)$ and the conclusion of the clause. As a result, the formula can be rewritten in the form

$$\Phi \coloneqq \exists \mathbf{R} \forall x \forall y \left[ \bigwedge_i C_i(x,y) \wedge \bigwedge_{i \in [1,k]} (\alpha_i(x,y) \to \theta_i(x,y)) \right]$$

where the $C_i$'s are the input clauses and the contradiction clauses, and each computation clause is written in the form $\alpha_i(x,y) \to \theta_i(x,y)$, where $\alpha_i(x,y)$ is a conjunction of formulas of the only forms $R(x-1,y) \wedge x > 1$, $R(x,y-1) \wedge y > 1$, and $\theta_i(x,y)$ is a Horn clause in which *all* atoms are of the form $R(x,y)$.

The second idea is to "solve" the Horn clauses $\theta_i$ according to the input clauses and *all the possible* conjunctions of hypotheses $\alpha_i$ that may be true. Notice the two following facts: the hypotheses of the input clauses are input literals and the conjuncts of the $\alpha_i$'s are of the only forms $R(x-1,y) \wedge x > 1$, $R(x,y-1) \wedge y > 1$. So, we can prove by induction on the sum $x+y$ that the obtained formula $\Phi'$ in which no atom $R(x,y)$ appears as a clause hypothesis, is equivalent to the above formula $\Phi$. The complete proof is given in Appendix A.

**Transformation of the formula into a grid-circuit.** Let $\mathbf{R} = \{R_1, \dots, R_m\}$ denote the set of binary predicates of the formula. By a case separation of the clauses, it is easy to transform the formula into an equivalent formula $\Phi \coloneqq \exists \mathbf{R} \forall x \forall y \psi$ where $\psi$ is a conjunction of clauses of the following forms (a-e), in which $s \in \Sigma$, $j \in [1,m]$, and $A, B$ are (possibly empty) subsets of $[1,m]$:

**(a)** $x = 1 \wedge y = 1 \wedge Q_s(y) \to R_j(x,y)$;
**(b)** $x = 1 \wedge y > 1 \wedge Q_s(y) \wedge \bigwedge_{i \in A} R_i(x,y-1) \to R_j(x,y)$;
**(c)** $x > 1 \wedge y = 1 \wedge \bigwedge_{i \in A} R_i(x-1,y) \to R_j(x,y)$;
**(d)** $x > 1 \wedge y > 1 \wedge \bigwedge_{i \in A} R_i(x-1,y) \wedge \bigwedge_{i \in B} R_i(x,y-1) \to R_j(x,y)$;
**(e)** $x = n \wedge y = n \wedge R_j(x,y) \to \bot$.

Now, transform this formula into a grid-circuit $\mathcal{C} \coloneqq (\Sigma, (\texttt{Input}_n)_{n>0}, \mathbf{Q}, \mathbf{Q}_{\mathsf{acc}}, \mathtt{g})$. The idea is that the state of a site $(x,y) \in [1,n]^2$ is the set of predicates $R_i$ such that $R_i(x,y)$ is true. Let $\mathbf{Q}$ be the power set of the set of $\mathbf{R}$ indices: $\mathbf{Q} \coloneqq \mathcal{P}([1,m])$. There are four types of transition (a-d) which mimic the clauses (a-d) above. These are, for $s \in \Sigma$ and $q, q' \in \mathbf{Q}$:

**(a)** $\mathtt{g}(\sharp, \sharp, s) = \{j \in [1,m] \mid$ there is a clause (a) with $Q_s$, and conclusion $R_j(x,y)\}$;
**(b)** $\mathtt{g}(q, \sharp, s) = \{j \in [1,m] \mid$ there is a clause (b) with $Q_s$, and $A \subseteq q$, and conclusion $R_j(x,y)\}$;
**(c)** $\mathtt{g}(\sharp, q, \$) = \{j \in [1,m] \mid$ there is a clause (c) with $A \subseteq q$, and conclusion $R_j(x,y)\}$;
**(d)** $\mathtt{g}(q, q', \$) = \{j \in [1,m] \mid \exists$ a clause (d) with $A \subseteq q$, $B \subseteq q'$, and conclusion $R_j(x,y)\}$.

Of course, the set of accepting states of $\mathcal{C}$ is determined by the contradiction clauses (e): $\mathbf{Q}_{\mathsf{acc}} \coloneqq \{q \in \mathbf{Q} \mid q$ contains no $j$ such that $R_j$ occurs in a clause (e)$\}$.

We can easily check the equivalence, for each $w \in \Sigma^+$: $\langle w \rangle \models \Phi \iff \mathcal{C}$ accepts $w$. Therefore, the inclusion $\texttt{pred-ESO-HORN} \subseteq \texttt{Grid}$ is proved. ◀

## 3.3 Grid-circuits are equivalent to real-time 1-CA

▶ **Lemma 14.** *[12]* $\texttt{Grid} = \texttt{RealTime1CA}$.

**Proof.** Figure 4 shows how `Grid` is simulated on `RealTime1CA` and Figure 5 shows how `RealTime1CA` is simulated on `Grid`. The proof is detailed in Appendix B. ◀

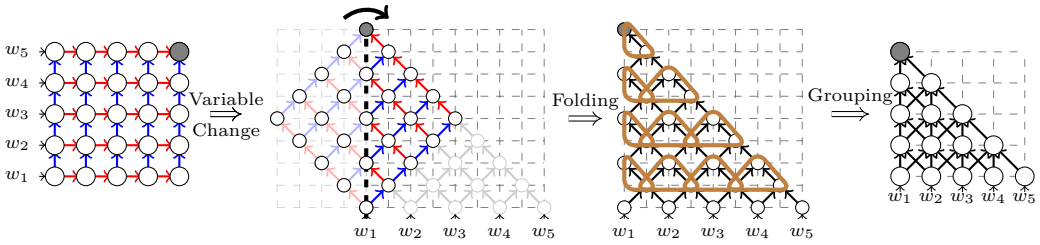



**Figure 4** Simulation of `Grid` on `RealTime1CA`.

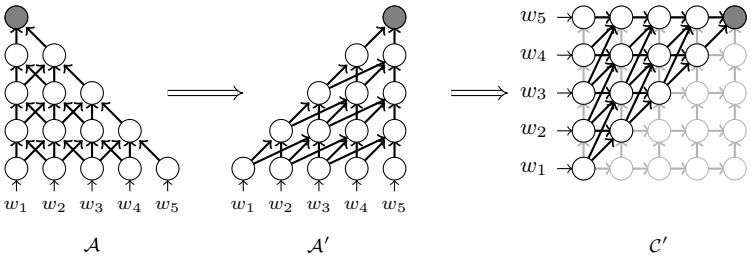



**Figure 5** Simulation of `RealTime1CA` on the grid-circuit.

**Proof of Theorem 8.** Lemmas 13 and 14 give us the following equalities of classes: `pred-ESO-HORN` = `Grid` = `RealTime1CA`. These equalities trivially hold when restricted to unary languages: $\texttt{pred-ESO-HORN}_1 = \texttt{Grid}_1 = \texttt{RealTime1CA}_1$.

From the fact that the class $\texttt{RealTime1CA}_1$ is closed under complement and from Lemma 10, we deduce $\texttt{Conj}_1 \subseteq \texttt{pred-ESO-HORN}_1 = \texttt{Grid}_1 = \texttt{RealTime1CA}_1$. ◀

## 4 Real-time recognition of a conjunctive language: the general case

Recall the inclusions[4] $\texttt{RealTime1CA} \subseteq \texttt{RealTime2OCA} \subseteq \texttt{RealTime2SOCA}$.

Our second main result strengthens the inclusion $\texttt{CFL} \subseteq \texttt{RealTime2SOCA}$ of Terrier [29]:

▶ **Theorem 15.** $\texttt{Conj} \subseteq \texttt{RealTime2OCA}$.

### 4.1 Expressing a conjunctive language in logic: the general case

The generating process of a conjunctive language is naturally expressed in the Horn logic `incl-pred-ESO-HORN`. This is a hybrid logic with three first-order variables $x, y, z$, whose name means that it makes inductions on the variable interval $[x, y]$, by *inclusion*, and on the individual variable $z$, by *predecessor*.

▶ **Definition 16** (`incl-pred-ESO-HORN`). *A formula of* `incl-pred-ESO-HORN` *is a formula* $\Phi := \exists \mathbf{R} \forall x \forall y \forall z \psi(x, y, z)$ *where* $\mathbf{R}$ *is a finite set of ternary predicates, and* $\psi$ *is a conjunction of Horn clauses, of signature*[5] $\mathcal{S}_\Sigma \cup \mathbf{R} \cup \{=, \leq\}$*, and of the three following forms:*

- *an* input clause: $x = y \wedge \texttt{min}(z) \wedge Q_s(x) \rightarrow R(x, y, z)$ *with* $s \in \Sigma$ *and* $R \in \mathbf{R}$*;*

[4] Recall that `RealTime2SOCA` is the class of languages recognized by *sequential* two-dimensional one-way cellular automata in real-time: this is the minimal time, $3n - 1$, for the output cell $(n, n)$ to receive the $n$ letters of the input word, communicated sequentially by the input cell $(1, 1)$.

[5] This definition must consider $=$ and $\leq$ as primitive symbols.

- *a* computation clause: $\delta_1 \wedge \ldots \wedge \delta_r \to R(x,y,z)$ *with* $R \in \mathbf{R}$ *and where each hypothesis* $\delta_h$ *is an atom* $S(x,y,z)$ *or a conjunction* $S(x+i, y-k, z-k) \wedge x+i \leq y-j \wedge z > k$ *with* $S \in \mathbf{R}$ *and* $i,j,k \geq 0$ *three integers such that* $i+j+k > 0$*;*
- *a* contradiction clause: $\texttt{min}(x) \wedge \texttt{max}(y) \wedge \texttt{max}(z) \wedge R(x,y,z) \to \bot$ *with* $R \in \mathbf{R}$.

Let us also call `incl-pred-ESO-HORN` the class of languages defined by a formula of `incl-pred-ESO-HORN`.

▶ **Lemma 17.** *For each language* $L \subseteq \Sigma^+$*, if* $L \in \texttt{Conj}$*, then* $\Sigma^+ \setminus L \in$ `incl-pred-ESO-HORN`.

**Proof.** The proof is a variation (an extension) of the proof of the same result, Lemma 10, in the unary case. This is why we insist on the differences. Let $G = (\Sigma, N, P, S)$ be a conjunctive grammar in binary normal form which generates $L$ and let $w$ be a word $w = w_1 \ldots w_n \in \Sigma^+$. For each $A \in N$ and each factor $w_{x,y} \coloneqq w_x \ldots w_y$, we have, according to the length $y-x+1$ of $w_{x,y}$, the following equivalences which will be the basis of our induction:

- if $x = y$, then $w_{x,y} \in L(A) \iff$ the short rule $A \to w_x$ belongs to $P$;
- if $x < y$, then $w_{x,y} \in L(A) \iff$ there is a long rule $A \to B_1C_1 \& \ldots \& B_mC_m$ in $P$ such that, for each $i \in \{1, \ldots, m\}$, there exists $z \geq \lceil (y-x+1)/2 \rceil$ such that either $w_{x,x+z-1} \in L(B_i)$ and $w_{x+z,y} \in L(C_i)$, or $w_{x,y-z} \in L(B_i)$ and $w_{y-z+1,y} \in L(C_i)$.

Thus, a double induction is performed, on the index interval $[x,y]$ of a factor $w_{x,y}$ and the maximal $z$ among the lengths of the two sub-factors $u, v$ of the $m$ decompositions $w_{x,y} = uv$, $u \in L(B_i)$, $v \in L(C_i)$, for a long rule. This is naturally expressed in the logic `incl-pred-ESO-HORN`.

We want to construct a first-order formula $\forall x \forall y \forall z \psi_G$ of signature $\mathcal{S}_\Sigma \cup \mathbf{R} \cup \{=, \leq\}$, for the set of *ternary* predicates $\mathbf{R} \coloneqq \{\texttt{Pref}_A^{\texttt{Maj}}, \texttt{Pref}_A^{\texttt{Min}}, \texttt{Suff}_A^{\texttt{Maj}}, \texttt{Suff}_A^{\texttt{Min}} \mid A \in N\} \cup \{\texttt{Concat}_{BC} \mid B, C \in N\} \cup \mathbf{R}_{\texttt{arith}}$, so that the formula $\Phi_G \coloneqq \exists \mathbf{R} \forall x \forall y \forall z \psi_G$ belongs to `incl-pred-ESO-HORN` and defines the language $\Sigma^+ \setminus L$. The intuitive meanings of the predicates $\texttt{Pref}_A^{\texttt{Maj}}, \texttt{Pref}_A^{\texttt{Min}}, \texttt{Suff}_A^{\texttt{Maj}}, \texttt{Suff}_A^{\texttt{Min}}$ and $\texttt{Concat}_{BC}$ are as follows:

- $\texttt{Pref}_A^{\texttt{Maj}}(x,y,z) \iff \left\lceil \frac{y-x+1}{2} \right\rceil \leq z \leq y-x+1$ and $w_{x,x+z-1} \in L(A)$;
- $\texttt{Pref}_A^{\texttt{Min}}(x,y,z) \iff \left\lceil \frac{y-x+1}{2} \right\rceil \leq z \leq y-x$ and $w_{x,y-z} \in L(A)$;
- $\texttt{Suff}_A^{\texttt{Maj}}(x,y,z) \iff \left\lceil \frac{y-x+1}{2} \right\rceil \leq z \leq y-x+1$ and $w_{y-z+1,y} \in L(A)$;
- $\texttt{Suff}_A^{\texttt{Min}}(x,y,z) \iff \left\lceil \frac{y-x+1}{2} \right\rceil \leq z \leq y-x$ and $w_{x+z,y} \in L(A)$;
- $\texttt{Concat}_{BC}(x,y,z) \iff$ there is some $z'$ with $\left\lceil \frac{y-x+1}{2} \right\rceil \leq z' \leq z$ such that either $w_{x,x+z'-1} \in L(B)$ and $w_{x+z',y} \in L(C)$, or $w_{x,y-z'} \in L(B)$ and $w_{y-z'+1,y} \in L(C)$.

Note that the above equivalences for $\texttt{Pref}_A^{\texttt{Maj}}$ and $\texttt{Suff}_A^{\texttt{Maj}}$ imply in the particular case $z = y-x+1$ the equivalences $\texttt{Pref}_A^{\texttt{Maj}}(x,y,z) \iff \texttt{Suff}_A^{\texttt{Maj}}(x,y,z) \iff w_{x,y} \in L(A)$.

Let us give and justify a list of Horn clauses whose conjunction $\psi'_G$ defines the predicates $\texttt{Pref}_A^{\texttt{Maj}}, \texttt{Pref}_A^{\texttt{Min}}, \texttt{Suff}_A^{\texttt{Maj}}, \texttt{Suff}_A^{\texttt{Min}}$ and $\texttt{Concat}_{BC}$, using the arithmetic predicates $z = y-x+1$, $y-x+1 = 2z$, and $z \geq \left\lceil \frac{y-x+1}{2} \right\rceil$ easily defined in `incl-pred-ESO-HORN`.

**Short rules.** Each rule $A \to s$ of $P$ is expressed by the two clauses:

- $x = y \wedge z = 1 \wedge Q_s(x) \to \texttt{Pref}_A^{\texttt{Maj}}(x,y,z)$ ; $x = y \wedge z = 1 \wedge Q_s(x) \to \texttt{Suff}_A^{\texttt{Maj}}(x,y,z)$.

**Induction for prefixes.** If we have for $x < y$ the inequalities $\left\lceil \frac{(y-1)-x+1}{2} \right\rceil \leq z \leq (y-1)-x+1$ and $z \geq \left\lceil \frac{y-x+1}{2} \right\rceil$ then $\left\lceil \frac{y-x+1}{2} \right\rceil \leq z \leq y-x+1$. This justifies the clause:

- $x \leq y-1 \wedge \texttt{Pref}_A^{\texttt{Maj}}(x,y-1,z) \wedge z \geq \left\lceil \frac{y-x+1}{2} \right\rceil \to \texttt{Pref}_A^{\texttt{Maj}}(x,y,z)$, for all $A \in N$.

For $x < y$ and $y-x+1 = 2z$, we have $w_{x,x+z-1} = w_{x,y-z}$ and $\left\lceil \frac{y-x+1}{2} \right\rceil \leq z \leq y-x$. This justifies the clause:

- $x \leq y-1 \wedge \texttt{Pref}_A^{\texttt{Maj}}(x,y-1,z) \wedge y-x+1 = 2z \to \texttt{Pref}_A^{\texttt{Min}}(x,y,z)$, for all $A \in N$.

For $x < y$ and $z > 1$ and $\left\lceil \frac{(y-1)-x+1}{2} \right\rceil \leq z-1 \leq (y-1)-x$, we have $\lceil \frac{y-x+1}{2} \rceil \leq z \leq y-x$. This justifies the clause:

- $x \leq y-1 \wedge z > 1 \wedge \mathtt{Pref}_A^{\mathtt{Min}}(x, y-1, z-1) \to \mathtt{Pref}_A^{\mathtt{Min}}(x, y, z)$, for all $A \in N$.

**Induction for suffixes.** As this induction is symmetric to the one for prefixes, we do not justify the following list of induction clauses for the predicates $\mathtt{Suff}_A^{\mathtt{Maj}}$ and $\mathtt{Suff}_A^{\mathtt{Min}}$, $A \in N$:

- $x+1 \leq y \wedge \mathtt{Suff}_A^{\mathtt{Maj}}(x+1, y, z) \wedge z \geq \lceil \frac{y-x+1}{2} \rceil \to \mathtt{Suff}_A^{\mathtt{Maj}}(x, y, z)$;
- $x+1 \leq y \wedge \mathtt{Suff}_A^{\mathtt{Maj}}(x+1, y, z) \wedge y-x+1 = 2z \to \mathtt{Suff}_A^{\mathtt{Min}}(x, y, z)$;
- $x+1 \leq y \wedge z > 1 \wedge \mathtt{Suff}_A^{\mathtt{Min}}(x+1, y, z-1) \to \mathtt{Suff}_A^{\mathtt{Min}}(x, y, z)$.

**Concatenation.** For all $B, C \in N$, it is clear that the concatenation predicate $\mathtt{Concat}_{BC}$ is defined inductively by the following three clauses:

- *initialization:* $\mathtt{Pref}_B^{\mathtt{Maj}}(x, y, z) \wedge \mathtt{Suff}_C^{\mathtt{Min}}(x, y, z) \to \mathtt{Concat}_{BC}(x, y, z)$;
  $\mathtt{Pref}_B^{\mathtt{Min}}(x, y, z) \wedge \mathtt{Suff}_C^{\mathtt{Maj}}(x, y, z) \to \mathtt{Concat}_{BC}(x, y, z)$;
- *induction:* $z > 1 \wedge \mathtt{Concat}_{BC}(x, y, z-1) \to \mathtt{Concat}_{BC}(x, y, z)$.

**Long rules.** Each rule $A \to B_1C_1 \& \ldots \& B_mC_m$ of $P$ is expressed by the two clauses:

- $z = y-x+1 \wedge \mathtt{Concat}_{B_1C_1}(x, y, z) \wedge \cdots \wedge \mathtt{Concat}_{B_mC_m}(x, y, z) \to \mathtt{Pref}_A^{\mathtt{Maj}}(x, y, z)$;
- $z = y-x+1 \wedge \mathtt{Concat}_{B_1C_1}(x, y, z) \wedge \cdots \wedge \mathtt{Concat}_{B_mC_m}(x, y, z) \to \mathtt{Suff}_A^{\mathtt{Maj}}(x, y, z)$.

Thus, the formula $\forall x \forall y \forall z \psi'_G$ where $\psi'_G$ is the conjunction of the above clauses defines the predicates $\mathtt{Pref}_A^{\mathtt{Maj}}, \mathtt{Pref}_A^{\mathtt{Min}}, \mathtt{Suff}_A^{\mathtt{Maj}}, \mathtt{Suff}_A^{\mathtt{Min}}$, and $\mathtt{Concat}_{BC}$.

**Definition of $\Sigma^+ \setminus L$.** We have the equivalence $\mathtt{Pref}_S^{\mathtt{Maj}}(1, n, n) \iff w \in L(S) \iff w \in L$. Therefore, the following contradiction clause expresses $w \notin L$:

- $\gamma_S \coloneqq \mathtt{min}(x) \wedge \mathtt{max}(y) \wedge \mathtt{max}(z) \wedge \mathtt{Pref}_S^{\mathtt{Maj}}(x, y, z) \to \bot$.

Finally, observe that the formula $\Phi_G \coloneqq \exists \mathbf{R} \forall x \forall y \forall z \psi_G$ where $\psi_G$ is $\gamma_{\mathtt{arith}} \wedge \psi'_G \wedge \gamma_S$ and $\gamma_{\mathtt{arith}}$ is the conjunction of clauses that define the arithmetic predicates, belongs to `incl-pred-ESO-HORN`. Since we have $\langle w \rangle \models \Phi_G \iff w \notin L$, as justified above, then the language $\Sigma^+ \setminus L$ belongs to `incl-pred-ESO-HORN`, as claimed. ◀

## 4.2 Equivalence of logic with cube-circuits

We now introduce the *cube-circuit*, an extension of the grid-circuit to three dimensions. It will make the link between our logic `incl-pred-ESO-HORN` and the class `RealTime2OCA`.

▶ **Definition 18.** *A* cube-circuit *is a tuple* $\mathcal{C} \coloneqq (\Sigma, (\mathtt{Input}_n)_{n>0}, \mathbf{Q}, \mathbf{Q}_{\mathtt{acc}}, \mathtt{g})$ *where*

- $\Sigma$ *is the* input alphabet *and* $(\mathtt{Input}_n)_{n>0}$ *is the family of* input functions $\mathtt{Input}_n : \Sigma^n \times [1,n]^3 \to \Sigma \cup \{\$\}$ *such that, for* $w = w_1 \ldots w_n \in \Sigma^n$, $\mathtt{Input}_n(w, x, y, z) = w_x$ *if* $x = y$ *and* $z = 1$, *and* $\mathtt{Input}_n(w, x, y, z) = \$$ *otherwise,*
- $\mathbf{Q} \cup \{\sharp\}$ *is the finite set of* states *and* $\mathbf{Q}_{\mathtt{acc}} \subseteq \mathbf{Q}$ *is the subset of* accepting states,
- $\mathtt{g} : (\mathbf{Q} \cup \{\sharp\})^3 \times (\Sigma \cup \{\$\}) \to \mathbf{Q}$ *is the* transition function.

▶ **Definition 19** (computation of a cube-circuit)**.** *The* computation $\mathcal{C}_w$ *of a cube-circuit* $\mathcal{C} \coloneqq (\Sigma, (\mathtt{Input}_n)_{n>0}, \mathbf{Q}, \mathbf{Q}_{\mathtt{acc}}, \mathtt{g})$ *on a word* $w = w_1 \ldots w_n \in \Sigma^n$ *is a grid of* $(n+1)^3$ *sites* $(x, y, z) \in [1, n+1] \times [0, n]^2$, *each in a state* $\langle x, y, z \rangle \in \mathbf{Q} \cup \{\sharp\}$ *computed inductively:*

- *each site* $(x, y, z)$ *such that* $x > y$ *or* $z = 0$ *is in the state* $\sharp$*;*
- *the state of each site* $(x, y, z) \in [1, n]^3$ *such that* $x \leq y$ *and* $z > 0$ *is* $\langle x, y \rangle = \mathtt{g}(\langle x+1, y, z \rangle, \langle x, y-1, z \rangle, \langle x, y, z-1 \rangle, \mathtt{Input}_n(w, x, y, z))$.

A word $w = w_1 \dots w_n \in \Sigma^n$ is *accepted* by the cube-circuit $\mathcal{C}$ if the output state $\langle 1, n, n \rangle$ of $\mathcal{C}_w$ belongs to $\mathbf{Q}_{\mathsf{acc}}$. The language *recognized* by $\mathcal{C}$ is the set of words it accepts. We denote by `Cube` the class of languages recognized by a cube-circuit.

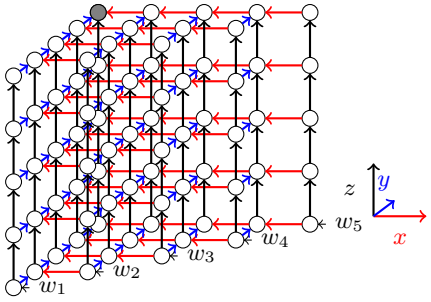



**Figure 6** The cube-circuit.

Actually, the logic `incl-pred-ESO-HORN` is equivalent to cube-circuits.

▶ **Lemma 20.** `incl-pred-ESO-HORN` = `Cube`.

**Proof.** The proof is similar to that of `pred-ESO-HORN` = `Grid` (Lemma 13). The cube-circuit can be seen as the "normalized form" of a formula of `incl-pred-ESO-HORN`, proving the inclusion `Cube` $\subseteq$ `incl-pred-ESO-HORN`. The proof of the inverse inclusion is divided into the same three steps as for Lemma 13, which must be adapted to three variables: 1) elimination of atoms $R(x+i, y-j, z-k)$ for $i+j+k > 1$ (instead of elimination of atoms $R(x-i, y-j)$ for $i+j > 1$); 2) elimination of hypotheses $R(x, y, z)$ (instead of elimination of hypotheses $R(x, y)$); 3) transformation of the resulting formula into a cube-circuit.

Steps 1 and 2 are adapted straightforwardly. Let us describe in detail step 3. Let $\mathbf{R} = \{R_1, \dots, R_m\}$ denote the set of ternary predicates of the formula resulting from step 2. By a case separation of the clauses, it is easy to transform this formula into an equivalent formula $\Phi := \exists \mathbf{R} \forall x \forall y \forall z \psi$ where $\psi$ is a conjunction of clauses of the following forms (a-e), in which $s \in \Sigma$, $j \in [1, m]$, and $A, B, C$ are (possibly empty) subsets of $[1, m]$:

**(a)** $x = y \wedge z = 1 \wedge Q_s(x) \to R_j(x, y, z)$;
**(b)** $x < y \wedge z = 1 \wedge \bigwedge_{i \in A} R_i(x+1, y, z) \wedge \bigwedge_{i \in B} R_i(x, y-1, z) \to R_j(x, y, z)$;
**(c)** $x = y \wedge z > 1 \wedge \bigwedge_{i \in A} R_i(x, y, z-1) \to R_j(x, y, z)$;
**(d)** $x < y \wedge z > 1 \wedge \bigwedge_{i \in A} R_i(x+1, y, z) \wedge \bigwedge_{i \in B} R_i(x, y-1, z) \wedge \bigwedge_{i \in C} R_i(x, y, z-1) \to R_j(x, y, z)$;
**(e)** $x = 1 \wedge y = n \wedge z = n \wedge R_j(x, y, z) \to \bot$.

Now, transform this formula into a cube-circuit $\mathcal{C} := (\Sigma, (\mathtt{Input}_n)_{n>0}, \mathbf{Q}, \mathbf{Q}_{\mathsf{acc}}, \mathsf{g})$. The idea is still that the state of a site $(x, y, z) \in [1, n]^3$ is the set of predicates $R_i$ such that $R_i(x, y, z)$ is true, and $\mathbf{Q}$ is again the power set of the set of $\mathbf{R}$ indices: $\mathbf{Q} := \mathcal{P}([1, m])$. There are four types of transition (a-d), which mimic the clauses (a-d) above. These are, for $s \in \Sigma$ and $q, q', q'' \in \mathbf{Q}$:

**(a)** $\mathsf{g}(\sharp, \sharp, \sharp, s) = \{j \in [1, m] \mid \exists$ a clause (a) with $Q_s$, and conclusion $R_j(x, y, z)\}$;
**(b)** $\mathsf{g}(q, q', \sharp, \$) = \{j \in [1, m] \mid \exists$ a clause (b) with $A \subseteq q$, $B \subseteq q'$, and conclusion $R_j(x, y, z)\}$;
**(c)** $\mathsf{g}(\sharp, \sharp, q, \$) = \{j \in [1, m] \mid \exists$ a clause (c) with $A \subseteq q$, and conclusion $R_j(x, y, z)\}$;
**(d)** $\mathsf{g}(q, q', q'', \$) = \{j \in [1, m] \mid \exists$ a clause (d) with $A \subseteq q$, $B \subseteq q'$, $C \subseteq q''$, and conclusion $R_j(x, y, z)\}$.

Here again, the set of accepting states of $\mathcal{C}$ is determined by the contradiction clauses (e): $\mathbf{Q}_{\mathsf{acc}} := \{q \in \mathbf{Q} \mid q$ contains no $j$ such that $R_j$ occurs in a clause (e)$\}$.

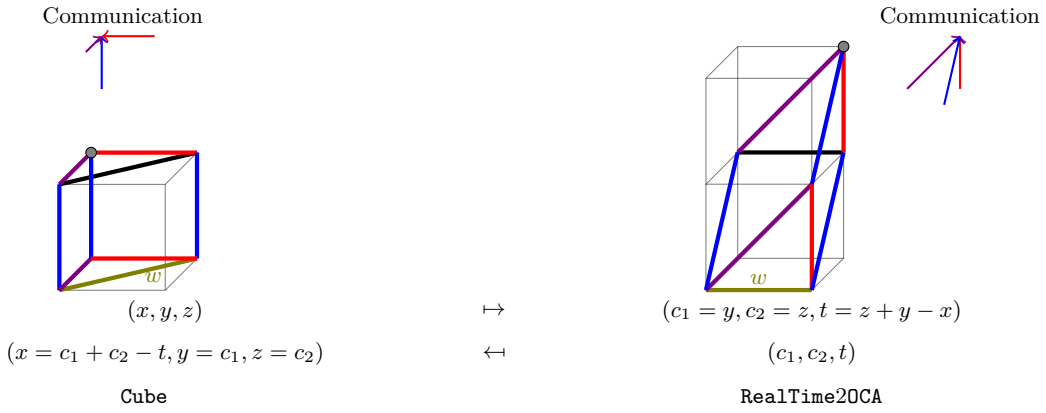



**Figure 7** Bijection between the sites of $\mathcal{C}_w$ and the space-time sites of a 2-OCA on $w$.

We can easily check the equivalence, for each $w \in \Sigma^+$: $\langle w \rangle \models \Phi \iff \mathcal{C}$ accepts $w$. Therefore, the inclusion incl-pred-ESO-HORN $\subseteq$ Cube is proved. ◀

## 4.3 Cube-circuits are equivalent to real-time 2-OCA

One observes that by a one-to-one transformation, the computation $\mathcal{C}_w$ of a cube-circuit $\mathcal{C}$ on a word $w$ is nothing else than the space-time diagram of a real-time 2-OCA on the input $w$. This yields:

▶ **Lemma 21.** Cube = RealTime2OCA.

**Proof.** The bijection between the sites $(x, y, z)$ of the computation $\mathcal{C}_w$ of a cube-circuit $\mathcal{C}$ on a word $w$ and the sites $(c_1, c_2, t)$ of the space-time diagram of a real-time 2-OCA on the input $w$ is depicted in Figure 7. We check that this bijection respects the communication scheme and the input/output sites of both computation models as shown in Figure 7. By this transformation, the transition function g of the cube-circuit, which is $\langle x, y, z \rangle = \mathtt{g}(\langle x+1, y, z \rangle, \langle x, y-1, z \rangle, \langle x, y, z-1 \rangle, \mathtt{Input}_n(w, x, y, z))$ becomes the transition function f of the 2-OCA: $\langle c_1, c_2, t \rangle = \mathtt{f}(\langle c_1, c_2, t-1 \rangle, \langle c_1 - 1, c_2, t-1 \rangle, \langle c_1, c_2 - 1, t-1 \rangle)$, and vice versa. ◀

**Proof of Theorem 15.** Lemmas 20 and 21 give us the following equalities of classes: incl-pred-ESO-HORN = Cube = RealTime2OCA.

From the fact that the class RealTime2OCA is closed under complement and from Lemma 17, we deduce Conj $\subseteq$ incl-pred-ESO-HORN = Cube = RealTime2OCA. ◀

# 5 Conclusion

We have proved the inclusions $\mathtt{Conj}_1 \subseteq$ RealTime1CA and Conj $\subseteq$ RealTime2OCA by expressing in two logics (proved equivalent to RealTime1CA and RealTime2OCA, respectively) the inductive process of a conjunctive grammar. These results contribute to a better knowledge of relationships between automata, grammars and logic. We think that they bring us closer to prove or disprove that Conj is a subclass of RealTime1CA.

Figure 8 recapitulates the known inclusions between the language classes that we have considered here. For each of the $\subseteq$ inclusions of this figure, whether it is strict or not is an open question. Note that it was necessary to add an extra dimension to the space-time

diagram to recognize any conjunctive language with a cellular automaton. Otherwise, any context-free or conjunctive language would always be decided by a RAM in time $O(n^2)$, which seems unlikely!

Besides, to grasp the expressive power, largely unknown, of the $\texttt{Conj}$ (resp. $\texttt{Conj}_1$) class, it would be important to obtain exact characterizations of this class in logic and/or computational complexity. This is a fascinating question for future research!

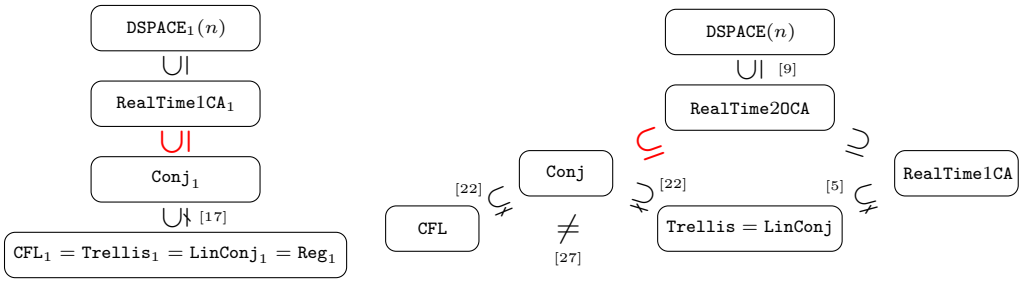



**Figure 8** Relations between language classes over a unary or general alphabet.

## References


1 Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

2 Nicolas Bacquey, Etienne Grandjean, and Frédéric Olive. Definability by Horn Formulas and Linear Time on Cellular Automata. In *ICALP 2017*, volume 80, pages 99:1–99:14, 2017.

3 Egon Börger, Erich Grädel, and Yuri Gurevich. *The Classical Decision Problem*. Perspectives in Mathematical Logic. Springer, 1997.

4 Jik H. Chang, Oscar H. Ibarra, and Michael A. Palis. Efficient simulations of simple models of parallel computation by time-bounded atms and space-bounded tms. *Theor. Comput. Sci.*, 68(1):19–36, 1989. doi:10.1016/0304-3975(89)90116-3.

5 Christian Choffrut and Karel Culik II. On real-time cellular automata and trellis automata. *Acta Informatica*, 21(4):393–407, 1984.

6 Marianne Delorme and Jacques Mazoyer. *Cellular Automata as Language Recognizers in Cellular Automata: a Parallel Model*. Kluwer, 1999.

7 Ronald Fagin. Generalized first-order spectra and polynomial-time recognizable sets. In *Complexity of Computation, SIAM-AMS Proceedings*, pages 43–73, 1974.

8 Dora Giammarresi, Antonio Restivo, Sebastian Seibert, and Wolfgang Thomas. Monadic second-order logic over rectangular pictures and recognizability by tiling systems. *Inf. Comput.*, 125(1):32–45, 1996. doi:10.1006/inco.1996.0018.

9 Leslie M. Goldschlager. A universal interconnection pattern for parallel computers. *J. ACM*, 29(4):1073–1086, 1982. doi:10.1145/322344.322353.

10 Erich Grädel. Capturing complexity classes by fragments of second-order logic. *Theoretical Computer Science*, 101(1):35–57, 1992.

11 Erich Grädel, Phokion G. Kolaitis, Leonid Libkin, Maarten Marx, Joel Spencer, Moshe Y. Vardi, Yde Venema, and Scott Weinstein. *Finite Model Theory and Its Applications*. Springer, 2007.

12 Etienne Grandjean and Théo Grente. Descriptive complexity for minimal time of cellular automata. In *LICS, 2019*, pages 1–13, 2019.

13 Etienne Grandjean, Théo Grente, and Véronique Terrier. Inductive definitions in logic versus programs of real-time cellular automata. *hal.archives-ouvertes.fr/hal-02474520/ submitted to Theoretical Computer Science*, 62 pages, February 2020.

14 Etienne Grandjean and Frédéric Olive. A logical approach to locality in pictures languages. *Journal of Computer and System Science*, 82(6):959–1006, 2016.

15 Oscar H. Ibarra and Tao Jiang. Relating the power of cellular arrays to their closure properties. *Theor. Comput. Sci.*, 57:225–238, 1988. doi:10.1016/0304-3975(88)90040-0.

16 Neil Immerman. *Descriptive complexity*. Springer, 1999.

17 Artur Jez. Conjunctive grammars generate non-regular unary languages. *Int. J. Found. Comput. Sci.*, 19(3):597–615, 2008. doi:10.1142/S012905410800584X.

18 K. N. King. Alternating multihead finite automata. *Theor. Comput. Sci.*, 61:149–174, 1988. doi:10.1016/0304-3975(88)90122-3.

19 Hans Kleine Büning and Theodor Lettmann. *Propositional logic - deduction and algorithms*, volume 48 of *Cambridge tracts in theoretical computer science*. Cambridge University Press, 1999.

20 S. Rao Kosaraju. Speed of recognition of context-free languages by array automata. *SIAM J. Comput.*, 4(3):331–340, 1975. doi:10.1137/0204028.

21 Leonid Libkin. *Elements of Finite Model Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2004.

22 Alexander Okhotin. Conjunctive grammars. *J. Autom. Lang. Comb.*, 6(4):519–535, 2001. doi:10.25596/jalc-2001-519.

23 Alexander Okhotin. Conjunctive grammars and systems of language equations. *Program. Comput. Softw.*, 28(5):243–249, 2002. doi:10.1023/A:1020213411126.

24 Alexander Okhotin. Boolean grammars. *Information and Computation*, 194(1):19–48, 2004.

25 Alexander Okhotin. On the equivalence of linear conjunctive grammars and trellis automata. *Theoretical Informatics and Applications*, 38(1):69–88, 2004.

26 Alexander Okhotin. Conjunctive and boolean grammars: The true general case of the context-free grammars. *Computer Science Review*, 9:27–59, 2013.

27 Véronique Terrier. Language not recognizable in real time by one-way cellular automata. *Theoretical Computer Science*, 156(1–2):281–287, March 1996.

28 Véronique Terrier. Closure properties of cellular automata. *Theor. Comput. Sci.*, 352(1-3):97–107, 2006. doi:10.1016/j.tcs.2005.10.039.

29 Véronique Terrier. Low complexity classes of multidimensional cellular automata. *Theor. Comput. Sci.*, 369(1-3):142–156, 2006.

30 Véronique Terrier. Language recognition by cellular automata. In *Handbook of Natural Computing*, pages 123–158. Springer, 2012.

31 Hao Wang. Dominoes and the aea case of the decision problem. In *Proceedings on the Symposium on the Mathematical Theory of Automata, April 1962*, pages 23–55, 1963.


## A Complement of proof for Lemma 13

**Elimination of hypotheses $R(x, y)$.** The first idea is to group together in each computation clause the hypothesis atoms of the form $R(x, y)$ and the conclusion of the clause. Accordingly, the formula obtained $\Phi$ can be rewritten in the form

$$\Phi := \exists \mathbf{R} \forall x \forall y \left[ \bigwedge_i C_i(x, y) \wedge \bigwedge_{i \in [1,k]} (\alpha_i(x, y) \to \theta_i(x, y)) \right]$$

where the $C_i$'s are the input clauses and the contradiction clause and each computation clause is written in the form $\alpha_i(x, y) \to \theta_i(x, y)$ where $\alpha_i(x, y)$ is a conjunction of formulas of the only forms $R(x-1, y) \wedge \neg \mathtt{min}(x)$, $R(x, y-1) \wedge \neg \mathtt{min}(y)$ (*but not* $R(x, y)$), and $\theta_i(x, y)$ is a Horn clause whose *all* atoms are of the form $R(x, y)$.

We number $R_1, \ldots, R_m$ the computation predicates of $\mathbf{R}$. To each subset $J \subseteq [1,k]$ of the family of implications $(\alpha_i(x,y) \to \theta_i(x,y))_{i\in[1,k]}$ let us associate the set

$$K_J \coloneqq \{h \in [1,m] \mid \bigwedge_{i\in J} \theta_i(x,y) \to R_h(x,y) \text{ is a tautology}\}.$$

Note that the notion of *tautology* used in the definition of $K_J$ is "propositional" because all the atoms involved are of the form $R_i(x,y)$, i.e., refer to the same pair of variables $(x,y)$. Also, note that the function $J \mapsto K_J$ is *monotonic*: for $J' \subseteq J$, we have $K_{J'} \subseteq K_J$ because $\bigwedge_{i\in J'} \theta_i(x,y) \to R_h(x,y)$ implies $\bigwedge_{i\in J} \theta_i(x,y) \to R_h(x,y)$.

Clearly, it is enough to prove the following claim:

▷ Claim 22. The formula $\Phi$ is equivalent to the following formula $\Phi'$, whose clauses have *no hypothesis* $R(x,y)$.

$$\Phi' \coloneqq \exists\mathbf{R}\forall x\forall y \left[\bigwedge_i C_i(x,y) \wedge \bigwedge_{J\subseteq[1,k]} \bigwedge_{h\in K_J} \left(\bigwedge_{i\in J} \alpha_i(x,y) \to R_h(x,y)\right)\right]$$

*Proof of the implication* $\Phi \Rightarrow \Phi'$: It is enough to prove the implication

$$\left[\bigwedge_{i\in[1,k]} (\alpha_i(x,y) \to \theta_i(x,y))\right] \to \left[\bigwedge_{i\in J} \alpha_i(x,y) \to \bigwedge_{h\in K_J} R_h(x,y)\right]$$

for all set $J \subseteq [1,k]$. The implication to be proved can be equivalently written:

$$\left[\bigwedge_{i\in J} \alpha_i(x,y) \wedge \bigwedge_{i\in[1,k]} (\alpha_i(x,y) \to \theta_i(x,y))\right] \to \bigwedge_{h\in K_J} R_h(x,y).$$

The sub-formula between brackets above implies the conjunction $\bigwedge_{i\in J} \theta_i(x,y)$. As the implication $\bigwedge_{i\in J} \theta_i(x,y) \to \bigwedge_{h\in K_J} R_h(x,y)$ is a tautology (by definition of $K_J$), the implication to be proved is a tautology too.

The converse implication $\Phi' \Rightarrow \Phi$ is more difficult to prove. It uses a folklore property of propositional Horn formulas easy to be proved:

▶ **Lemma 23** (Horn property: folklore)**.** *Let $F$ be a* strict Horn formula *of propositional calculus, that is a conjunction of clauses of the form $p_1 \wedge \ldots \wedge p_k \to p_0$ where $k \geq 0$ and the $p_i$'s are propositional variables. Let $F'$ be the conjunction of propositional variables $q$ such that the implication $F \to q$ is a tautology. $F$ has the same minimal model* [6] *as $F'$.*

*Proof of the implication* $\Phi' \Rightarrow \Phi$: Let $\langle w\rangle$ be a model of $\Phi'$ and let $(\langle w\rangle, \mathbf{R})$ be the minimal model of the Horn formula

$$\varphi' \coloneqq \forall x\forall y \left[\bigwedge_i C_i(x,y) \wedge \bigwedge_{J\subseteq[1,k]} \bigwedge_{h\in K_J} \left(\bigwedge_{i\in J} \alpha_i(x,y) \to R_h(x,y)\right)\right].$$

[6] For example, for $F \coloneqq p_1 \wedge p_3 \wedge (p_1 \wedge p_3 \to p_5) \wedge (p_1 \wedge p_2 \to p_4)$, we have $F' \coloneqq p_1 \wedge p_3 \wedge p_5$, which has the same minimal model $I$ as $F$; this model is given by $I(p_1) = I(p_3) = I(p_5) = 1$ and $I(p_2) = I(p_4) = 0$.

It is enough to show that $(\langle w\rangle, \mathbf{R})$ also satisfies the formula

$$\varphi \coloneqq \forall x \forall y \left[ \bigwedge_i C_i(x,y) \wedge \bigwedge_{i\in[1,k]} (\alpha_i(x,y) \to \theta_i(x,y)) \right].$$

As each $\alpha_i$ is a conjunction of formulas of the form $R(x-1,y) \wedge \neg\texttt{min}(x)$, or $R(x,y-1) \wedge \neg\texttt{min}(y)$, we make an induction on the domain $\{(a,b) \in [1,n]^2 \mid a+b \leq t\}$, for $t \in [1,2n]$. More precisely, we are going to prove, by recurrence on the integer $t \in [1,2n]$, that the minimal model $(\langle w\rangle, \mathbf{R})$ of $\varphi'$ satisfies the "relativized" formula $\varphi_t$ of the formula $\varphi$ defined by

$$\varphi_t \coloneqq \forall x \forall y \left[ x+y \leq t \to \left[ \bigwedge_i C_i(x,y) \wedge \bigwedge_{i\in[1,k]} (\alpha_i(x,y) \to \theta_i(x,y)) \right] \right]$$

As the hypothesis $x+y \leq 2n$ holds for all $x, y$ in the domain $[1,n]$, $\varphi_{2n}$ is equivalent to $\varphi$ on the structure $(\langle w\rangle, \mathbf{R})$.

*Basis case:* For $t=1$ the set $\{(a,b) \in [1,n]^2 \mid a+b \leq t\}$ is empty so that the "relativized" formula $\varphi_1$ is trivially true in the minimal model $(\langle w\rangle, \mathbf{R})$ of $\varphi'$.

*Recurrence step:* Suppose $(\langle w\rangle, \mathbf{R}) \models \varphi_{t-1}$, for an integer $t \in [2,2n]$. It is enough to show that, for each couple $(a,b) \in [1,n]^2$ such that $a+b=t$, we have $(\langle w\rangle, \mathbf{R}) \models \bigwedge_{i\in[1,k]}(\alpha_i(a,b) \to \theta_i(a,b))$. Let $J_{a,b}$ be the set of indices $i \in [1,k]$ such that the couple $(a,b)$ satisfies $\alpha_i$:

$$J_{a,b} \coloneqq \{i \in [1,k] \mid (\langle w\rangle, \mathbf{R}) \models \alpha_i(a,b)\}.$$

Recall that each $\alpha_i(a,b)$ is a (possibly empty) conjunction of atoms $R(a',b')$ with $(a',b') = (a-1,b)$ or $(a',b') = (a,b-1)$, therefore such that $a'+b' = t-1$. Let $J \subseteq [1,k]$ be any set. Let us examine the two possible cases:

1) $J \subseteq J_{a,b}$: then the conjunction $\bigwedge_{i\in J} \alpha_i(a,b)$ holds in $(\langle w\rangle, \mathbf{R})$; hence, in $(\langle w\rangle, \mathbf{R})$, the conjunction $\bigwedge_{h\in K_J}(\bigwedge_{i\in J} \alpha_i(a,b) \to R_h(a,b))$ is equivalent to $\bigwedge_{h\in K_J} R_h(a,b)$;

2) $J \setminus J_{a,b} \neq \emptyset$: then the conjunction $\bigwedge_{i\in J} \alpha_i(a,b)$ is false in $(\langle w\rangle, \mathbf{R})$; hence, the conjunction $\bigwedge_{h\in K_J}(\bigwedge_{i\in J} \alpha_i(a,b) \to R_h(a,b))$ holds in $(\langle w\rangle, \mathbf{R})$.

From (1) and (2), we deduce that in $(\langle w\rangle, \mathbf{R})$ the conjunction $\bigwedge_{J\subseteq[1,k]} \bigwedge_{h\in K_J}(\bigwedge_{i\in J} \alpha_i(a,b) \to R_h(a,b))$ is equivalent to the conjunction $\bigwedge_{J\subseteq J_{a,b}} \bigwedge_{h\in K_J} R_h(a,b)$, which can be simplified as $\bigwedge_{h\in K_{J_{a,b}}} R_h(a,b)$ because $J \subseteq J_{a,b}$ implies $K_J \subseteq K_{J_{a,b}}$. Consequently, for all $h \in [1,m]$, the minimal model $(\langle w\rangle, \mathbf{R})$ of the Horn formula $\varphi'$ satisfies the atom $R_h(a,b)$ *iff* $h$ belongs to $K_{J_{a,b}}$. By definition,

$$K_{J_{a,b}} \coloneqq \{h \in [1,m] \mid \bigwedge_{i\in J_{a,b}} \theta_i(x,y) \to R_h(x,y) \text{ is a tautology}\}$$

or, equivalently,

$$K_{J_{a,b}} \coloneqq \{h \in [1,m] \mid \bigwedge_{i\in J_{a,b}} \theta_i(a,b) \to R_h(a,b) \text{ is a tautology}\}.$$

As a consequence of Lemma 23, the two conjunctions

$$\bigwedge_{i\in J_{a,b}} \theta_i(a,b) \text{ and } \bigwedge_{h\in K_{J_{a,b}}} R_h(a,b)$$

have the same minimal model, which is also the restriction of the minimal model $(\langle w\rangle, \mathbf{R})$ of $\varphi'$ to the set of atoms $R_h(a,b)$, for $h \in [1,m]$. Therefore, if $i \in J_{a,b}$, then $(\langle w\rangle, \mathbf{R}) \models \theta_i(a,b)$. If $i \in [1,k] \setminus J_{a,b}$, then we have $(\langle w\rangle, \mathbf{R}) \models \neg\alpha_i(a,b)$, by definition of $J_{a,b}$. Therefore, for all $i \in [1,k]$, we get $(\langle w\rangle, \mathbf{R}) \models \neg\alpha_i(a,b) \vee \theta_i(a,b)$. In other words, for all $(a,b)$ such that $a+b=t$, we have : $(\langle w\rangle, \mathbf{R}) \models \bigwedge_{i\in[1,k]}(\alpha_i(a,b) \to \theta_i(a,b))$ and then $(\langle w\rangle, \mathbf{R}) \models \varphi_t$.

This concludes the inductive proof that $(\langle w\rangle, \mathbf{R}) \models \varphi_t$, for all $t \in [1,2n]$, and then $\langle w\rangle \models \Phi$. This proves the converse implication $\Phi' \Rightarrow \Phi$. Claim 22 is demonstrated. □

## B Complement of proof for Lemma 14

**`Grid` $\subseteq$ `RealTime1CA`.** To prove this inclusion, we show how to simulate the computation of the grid-circuit on a real-time CA. The simulation is made by a geometric transformation that embeds the grid-circuit in the space-time diagram of a real-time CA. This transformation is divided into three steps:

1. a variable change: we apply to each site $(x,y) \in [1,n]^2$ of the grid-circuit the variable change $(x,y) \mapsto (c' = y - x + 1, t' = x + y - 1)$;
2. a folding: we fold the resulting diagram along the axis $c' = 1$: each site $(c',t')$ with $c' < 1$ is send to its symmetric counterpart $(-c'+1, t')$;
3. a grouping: each site $(c,t) = (\lceil \frac{c'}{2} \rceil, \lceil \frac{t'}{2} \rceil)$ of the new diagram records the set of sites $\{(c'-1,t'-1),(c',t'),(c'+1,t'-1)\}$ with $c'$ and $t'$ odd and greater than 1.

The resulting diagram is the expected space-time diagram of a real-time CA, proving the inclusion.

**`RealTime1CA` $\subseteq$ `Grid`.** To simulate a real-time CA $\mathcal{A} = (\mathtt{S}, \mathtt{S}_{accept}, \{-1,0,1\}, \mathtt{f})$ on the grid, we first turn $\mathcal{A}$ into an equivalent CA $\mathcal{A}' = (\mathtt{S}, \mathtt{S}_{accept}, \{-2,-1,0\}, \mathtt{f})$. This transformation can be seen as the variable change $(c,t) \mapsto (c+t-1, t)$. The diagram of $\mathcal{A}'$ is then embedded on the grid-circuit $\mathcal{C}'$ by applying to its sites $(c',t')$ the variable change $(c',t') \mapsto (t',c')$. The local and uniform communication of the embedded diagram can easily be carried out by the grid-circuit communication scheme.

# An Effective Construction for Cut-And-Project Rhombus Tilings with Global $n$-Fold Rotational Symmetry

**Victor H. Lutfalla**
Laboratoire d'Informatique de Paris Nord, Université Sorbonne Paris Nord, France

## Abstract

We give an explicit and effective construction for rhombus cut-and-project tilings with global $n$-fold rotational symmetry for any $n$. This construction is based on the dualization of regular $n$-fold multigrids. The main point is to prove the regularity of these multigrids, for this we use a result on trigonometric diophantine equations. A SageMath program that computes these tilings and outputs svg files is publicly available in [7].

**2012 ACM Subject Classification** Mathematics of computing → Discrete mathematics; Mathematics of computing → Combinatorics

**Keywords and phrases** Cut-and-project tiling, Rhombus tiling, Aperiodic order, Rotational symmetry, De Bruijn multigrid, Trigonometric diophantine equations

**Digital Object Identifier** 10.4230/OASIcs.AUTOMATA.2021.9

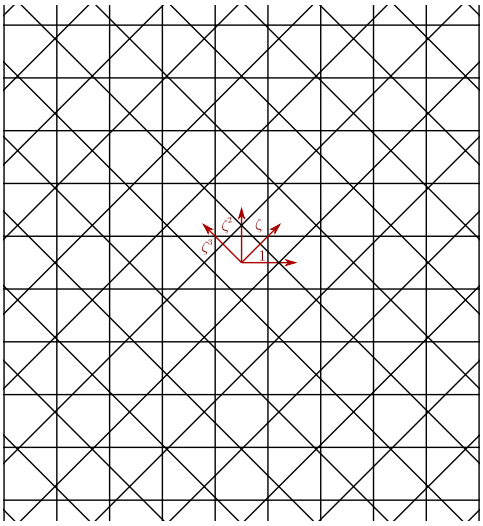
**Supplementary Material** n-fold multigrid dual tilings [7]
*Software*: `10.5281/zenodo.4698387`

**Acknowledgements** I want to thank Emmanuel Jeandel for pointing me towards the article *Trigonometric diophantine equations (On vanishing sums of roots of unit)* [3] when I presented him with the problem of regularity of multigrids. I also want to thank my advisor Thomas Fernique and my colleagues Lionel Pournin and Thierry Monteil for their help and advice.

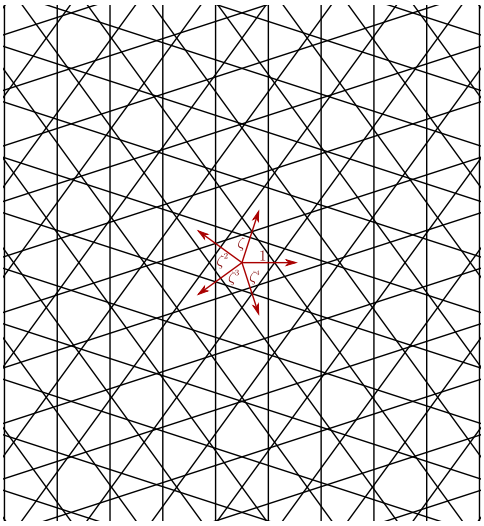
## 1 Introduction

In the 70s Penrose presented a quasiperiodic rhombus tiling with global 5-fold rotational symmetry and local 10-fold rotational symmetry [8]. This tiling is one of the most thoroughly studied tilings, for more details on the class of Penrose rhombus tilings and on the canonical Penrose rhombus tilings see [9, 1]. In 1981 de Bruijn proposed an algebraic definition of this tiling by the dualization of a pentagrid [4]. Later on a generalized multigrid method has been shown to be equivalent to the cut-and-project method or projection method for the construction of tilings by Gähler and Rhyner [5]. Here we use this dualization of multigrid method to give an effective construction of cut-and-project rhombus tilings with global $2n$-fold rotational symmetry for any $n$, and a similar construction with global $n$-fold rotational symmetry for odd $n$.

The main point of our work is to prove the regularity of the multigrids involved because when the multigrid is not regular its dual tiling contains tiles that are not rhombuses as depicted in Figure 3. Note that, by a non-constructive cardinality argument, the existence of such tilings has been known since the multigrid method was introduced and even though some might consider it known as *folk* we were not able to find any reference for explicit construction of regular grids with global $n$-fold rotational symmetry outside of the case of pentagrids [4] and tetragrids [2]. This explicit construction is used as an intermediate step for the construction of substitution discrete planes with $n$-fold rotational symmetry in [6], in that construction some regions of the multigrid dual tilings studied here are used to define substitution metatiles.



**(a)** $G_4(\frac{1}{2})$.



**(b)** $G_5(\frac{1}{2})$.

**Figure 1** Examples of multigrids.

Let us now give the relevant definitions for rhombus tilings and for the multigrid method before stating the main results.

A *rhombus tiling* is a covering of the plane without overlap by a set of rhombus tiles. Here we actually consider rhombus tilings of the complex plane $\mathbb{C}$ and we only consider the case where the set of tiles is finite up to translation and where the tiling is *edge-to-edge i.e.* any two tiles in the tiling either do not intersect, intersect on a single common vertex or intersect along a full common edge. A *patch* is a finite simply-connected set of non-overlapping tiles and a *pattern* is a patch up to translation.

A tiling is called *periodic* of period $\vec{v} \neq \vec{0}$ when it is invariant under the translation of vector $\vec{v}$ and *non-periodic* when it admits no period. A tiling is called *uniformly recurrent* or *uniformly repetitive* when for any pattern $p$ that appears in the tiling, there exists an "appearance" radius $R$ such that in the intersection of any open ball of radius $R$ with the tiling there is an occurrence of the pattern $p$. A tiling is called *quasiperiodic* when it is both non-periodic and uniformly recurrent.

A tiling is said to have *global $n$-fold rotational symmetry* when there exists a point $z$, usually the origin, such that the tiling is invariant under the rotation of center $z$ and angle $\frac{2\pi}{n}$. The *crystallographic restriction* theorem states that the only rotational symmetries possible for periodic tilings are 2-fold, 3-fold, 4-fold and 6-fold. Hence a tiling that has $n$-fold rotational symmetry for $n \notin \{2, 3, 4, 6\}$ is non-periodic.

*Cut-and-project* tilings are tilings that can be seen as the projection of a discrete plane of some $\mathbb{R}^n$ to the plane. The tiles of a cut-and-project tiling are the projection of the facets that form the corresponding discrete plane. We will not give here a precise definition since we will not use it later, for precise definitions see [1] or [9]. Note that cut-and-project tilings are used to model quasicrystals which means that they are not only of interest to mathematicians and theoretical computer-scientists but also to physicists and crystallographists. The only result we will use on cut-and-project tilings is that they are uniformly recurrent [9], in particular this implies that cut-and-project tilings with $n$-fold rotational symmetry for $n \notin \{2, 3, 4, 6\}$ are quasiperiodic.

Let us now define grids, multigrids and their dual tilings. The *grid* of direction $\xi \in \mathbb{C}$ with $|\xi| = 1$ and offset $\gamma \in \mathbb{R}$, denoted by $H(\xi, \gamma)$, is the set of equidistant lines orthogonal to $\xi$ and with offset $\gamma$ from the origin

$$H(\xi, \gamma) := \left\{ z \in \mathbb{C} \mid \mathrm{Re}\left(z \cdot \bar{\xi}\right) - \gamma \in \mathbb{Z} \right\}.$$

In this definition the important offset is actually the fractional part of $\gamma$ so we restrict the definition to the case $0 \leqslant \gamma < 1$. The *multigrid* of pairwise non-collinear directions $\xi = (\xi_i)_{0 \leqslant i < n}$ and offsets $\gamma = (\gamma_i)_{0 \leqslant i < n}$, denoted by $G_\xi(\gamma)$ is the union of the grids

$$G_\xi(\gamma) := \bigcup_{0 \leqslant i < n} H(\xi_i, \gamma_i).$$

For an integer $n \geqslant 3$ we define the $n$-fold direction $\zeta_n$ as

$$\zeta_n = \begin{cases} e^{\mathrm{i}\frac{2\pi}{n}} \text{ if } n \text{ is odd} \\ e^{\mathrm{i}\frac{\pi}{n}} \text{ if } n \text{ is even} \end{cases}$$

The *$n$-fold multigrid* of offsets $\gamma = (\gamma_i)_{0 \leqslant i < n}$ denoted $G_n(\gamma)$ is the multigrid with directions $(\zeta_n^i)_{0 \leqslant i < n}$

$$G_n(\gamma) := \bigcup_{0 \leqslant i < n} H(\zeta_n^i, \gamma_i)$$

For simplicity for some real number $x \in [0, 1[$ we denote $G_n(x)$ the multigrid with all offsets equal to $x$ *i.e.*

$$G_n(x) := G_n(x, \dots x) = \bigcup_{0 \leqslant i < n} H(\zeta_n^i, x)$$

See Figure 1 for a picture of $G_4(\frac{1}{2})$ and $G_5(\frac{1}{2})$.

From a multigrid $G_\xi(\gamma)$ we can define a tiling $P_\xi(\gamma)$ by a duality process. This tiling is dual to the multigrid in the sense that the multigrid, seen as a graph, is the adjacency graph of the tiling, see Figure 2 for an example. We will define this tiling only in the $n$-fold case, but one can easily adapt it to the general case by replacing the directions $\zeta_n^i$ by $\xi_i$.

Given a $n$-fold multigrid $G_n(\gamma)$ we define two functions $K : \mathbb{C} \to \mathbb{Z}^n$ and $f : \mathbb{C} \to \mathbb{C}$ as

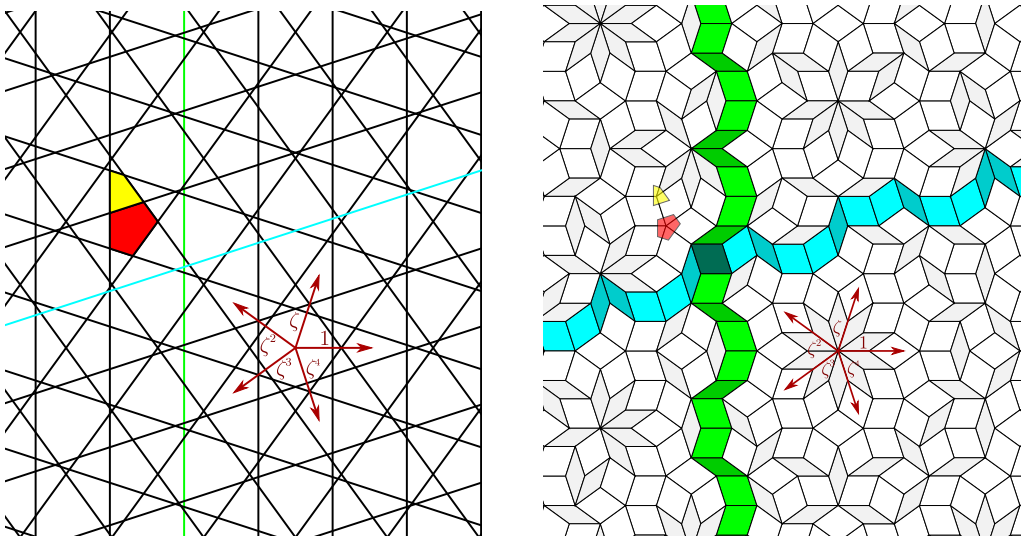
$$K(z) := \left( \left\lceil \mathrm{Re}\left(z \cdot \bar{\zeta_n^i}\right) - \gamma_i \right\rceil \right)_{0 \leqslant i < n} \qquad f(z) := \sum_{i=0}^{n-1} \left\lceil \mathrm{Re}\left(z \cdot \bar{\zeta_n^i}\right) - \gamma_i \right\rceil \zeta_n^i.$$

Remark that the functions $K$ and $f$ are constant on the interior of each cell, also called mesh, of the multigrid, so $f$ sends a cell of the multigrid to a single vertex.

The *multigrid dual tiling* of a $n$-fold multigrid $G_n(\gamma)$, denoted by $P_n(\gamma)$, is defined by its set of vertices $V$ and of edges $E$ as

$$V := f(\mathbb{C}) \qquad E := \left\{ \{z, z'\},\ z, z' \in V |\ \exists i, z' = z + \zeta_n^i \right\}$$

In this dualization process each cell or mesh of the multigrid is sent to a vertex of the dual tiling, see for example the cells in red and yellow and their dual in Figure 2, and each intersection point of the multigrid is sent to a tile of the dual tiling. The dual of an intersection point where $k$ lines intersect is a $2k$-gon with unit sides as shown in Figure 3 for the case of 5-fold multigrids. The dual of a line of the multigrid is a chain or ribbon of tiles that share an edge, see for example the green and blue line and their dual in Figure 2. Recall that multigrid dual tilings are cut-and-project tilings [9, 5, 1].




**(a)** The 5-fold multigrid or pentagrid $G_5(\frac{1}{2})$. **(b)** The dual tiling $P_5(\frac{1}{2})$.

**Figure 2** Example of a regular grid and its dual tiling, some elements of the multigrid and their dual in the tiling have been colored.

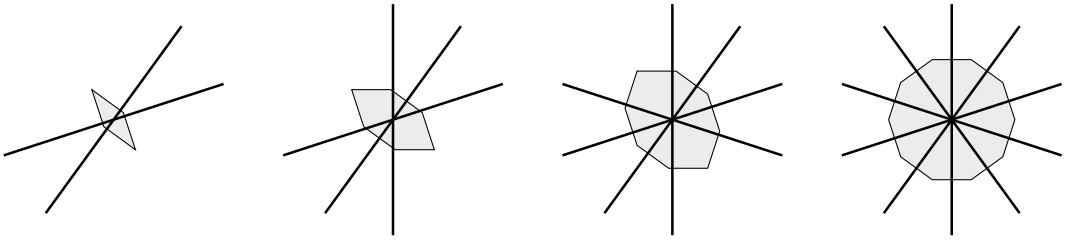


**Figure 3** Some possible intersection points in $G_5(\gamma)$ and their dual tiles.

A multigrid is called *singular* when there is at least one intersection point where at least 3 lines intersect, and is called *regular* otherwise. By definition, the dual tilings of regular multigrids are edge-to-edge rhombus tilings. For an example of a regular multigrid and its dual tiling, see Figure 2 where some elements of the multigrid and their dual are highlighted to emphasize the dualization process.

Now that we have defined all relevant terms, we can state the main result.

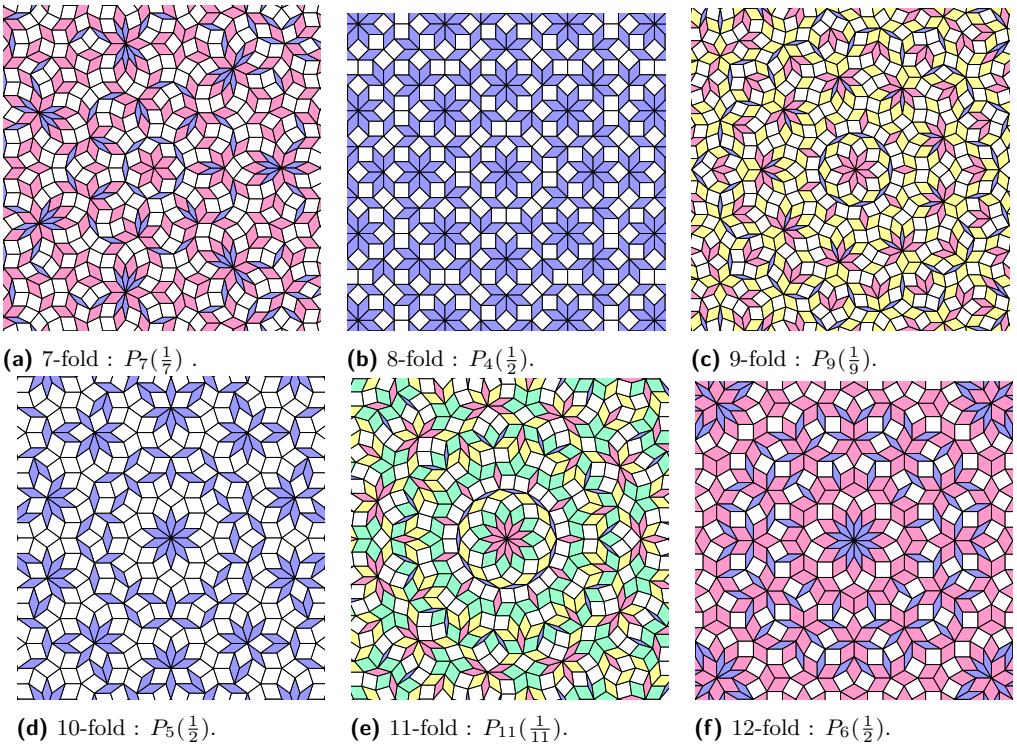
▶ **Theorem 1.**

1. *For any integer $n \geqslant 4$, the $n$-fold multigrid dual tiling $P_n(\frac{1}{2})$ is a cut-and-project quasiperiodic edge-to-edge rhombus tiling with global $2n$-fold rotational symmetry.*
2. *For any **odd** integer $n \geqslant 5$, the $n$-fold multigrid dual tiling $P_n(\frac{1}{n})$ is a cut-and-project quasiperiodic edge-to-edge rhombus tiling with global $n$-fold rotational symmetry.*

Theorem 1 is actually a corollary of Theorem 2 on the regularity of $n$-fold multigrids. See Figures 4 and 6 for examples of tilings $P_n(\frac{1}{2})$ and $P_n(\frac{1}{n})$, these figures have been produced using a SageMath program which is publicly available in [7].

▶ **Theorem 2.**

1. *For any $n \geqslant 3$ and any non-zero rational offset $r \in \mathbb{Q} \cap ]0, 1[$ the $n$-fold multigrid $G_n(r)$ is regular.*
2. *For any **odd** $n \geqslant 3$ and any tuple of non-zero rational offsets $\gamma = (\gamma_i)_{0 \leqslant i < n} \in (\mathbb{Q} \cap ]0, 1[)^n$ the $n$-fold multigrid $G_n(\gamma)$ is regular.*



**(a)** 7-fold : $P_7(\frac{1}{7})$ . **(b)** 8-fold : $P_4(\frac{1}{2})$. **(c)** 9-fold : $P_9(\frac{1}{9})$.

**(d)** 10-fold : $P_5(\frac{1}{2})$. **(e)** 11-fold : $P_{11}(\frac{1}{11})$. **(f)** 12-fold : $P_6(\frac{1}{2})$.

**Figure 4** Central patch of the multigrid dual tiling with exactly $n$-fold rotational symmetry for $n \in \{7, 8, 9, 10, 11, 12\}$.

Remark that Theorem 2 is not an exact characterization of regular $n$-fold multigrid but rather an easily checked sufficient condition for regularity. In [4] N. G. de Bruijn gives an exact characterization of regular pentagrids in the specific Penrose case (sum of the offsets is an integer), but this characterization is not easily generalized to the non-Penrose case and to all $n$-fold multigrids.

In Section 2 we show how Theorem 1 is a corollary of Theorem 2. In Section 3 we present the link between the regularity of multigrids and some trigonometric equations. In Section 4 we present the results of Conway and Jones on trigonometric diophantine equations [3]. In Section 5 we combine the results of Sections 3 and 4 to prove Theorem 2.

## 2 From regular $n$-fold multigrids to tilings with global $n$-fold symmetry

Let $n \geqslant 3$ be an integer. By Theorem 2 the multigrid $G_n(\frac{1}{2})$ and $G_n(\frac{1}{n})$ are regular, so their dual tilings $P_n(\frac{1}{2})$ and $P_n(\frac{1}{n})$ are edge-to-edge rhombus tilings. As mentioned above, the multigrid dual tilings are cut-and-project [5] and therefore also uniformly recurrent.

Let us remark that the dualization process commutes with rotations around the origin, so if a multigrid has some rotational symmetry around the origin then so does its dual tiling.

So for odd $n$, $P_n(\frac{1}{n})$ has global $n$-fold rotational symmetry because the grid also has global $n$-fold rotational symmetry, indeed applying the rotation of angle $\frac{2\pi}{n}$ centered on the origin to the multigrid sends $\zeta_n^k$ to $\zeta_n^{k+1}$ and since the offset is the same on all directions the rotated multigrid is the same as the original one. However this does not hold for even $n$ since in that case we chose $\zeta_n = e^{\mathrm{i}\frac{\pi}{n}}$ so we have $\zeta_n^n = e^{\mathrm{i}\pi} = -1 = -\zeta_n^0$ so the grid of offset $\frac{1}{n}$ for even $n \geqslant 4$ does not have any rotational symmetry.

Also for odd $n$, $P_n(\frac{1}{2})$ has global $2n$-fold rotational symmetry because the image of $\zeta_n^0$ by the rotation of angle $\frac{\pi}{n}$ is

$$e^{\mathrm{i}\frac{\pi}{n}} = -e^{\mathrm{i}\frac{(n+1)\pi}{n}} = -e^{\mathrm{i}\frac{2\lceil\frac{n}{2}\rceil\pi}{n}} = -\zeta_n^{\lceil\frac{n}{2}\rceil}$$

so with $1-\frac{1}{2}=\frac{1}{2}$ (*i.e.* the offset in direction $\zeta_n^i$ and in its reverse direction $-\zeta_n^i$ is the same) we get global $2n$-fold rotational symmetry for $G_n(\frac{1}{2})$ and $P_n(\frac{1}{2})$. For even $n$ we use also the fact that with offset $\frac{1}{2}$ we get offset along $\zeta_n^i$ is the same as along the opposite direction $-\zeta_n^i$ together with $\zeta_n = e^{\mathrm{i}\frac{\pi}{n}}$ which means that $\zeta_n^{n+i} = -\zeta_n^i$ to get global $2n$-fold rotational symmetry for $G_n(\frac{1}{2})$ and $P_n(\frac{1}{2})$.

When we combine these with the crystallographic restriction which implie that for any $n \geqslant 4$ these tilings are non-periodic we get Theorem 1. Remark that for even $n$, $P_n(\frac{1}{2})$ has global $2n$-fold rotational symmetry and $P_{\frac{n}{2}}(\frac{1}{2})$ has exactly $n$-fold global rotational symmetry. So for any $n$ there exists a tiling with exactly $n$-fold global rotational symetry, see the examples for $n \in \{7,8,9,10,11,12\}$ in Figure 4, and for $n=23$ in Figure 6.

Remark also that for odd $n$, for any $r \in \mathbb{Q} \cap ]0,1[$ the multigrid $G_n(r)$ is regular and the multigrid dual tiling $P_n(r)$ has global $n$-fold rotational symmetry, except the specific case $r=\frac{1}{2}$ that has global $2n$-fold rotational symmetry. The choice of $r=\frac{1}{n}$ is mainly due to the fact that the canonical Penrose rhombus tiling is $P_5(\frac{1}{5})$ so $P_n(\frac{1}{n})$ is in that sense a generalization of the canonical Penrose rhombus tiling.

Note that Theorem 1 is stated for $n>3$ because for $n=3$ the multigrid dual tilings with offset $\frac{1}{2}$ and $\frac{1}{3}$ are periodic.

## 3 Regularity of $n$-fold multigrids and trigonometric equations

In this section we present the link between the regularity or singularity of $n$-fold multigrids and some trigonometric equations.

▶ **Proposition 3** (Regularity of multigrids and trigonometric equations)**.** *Let $n \in \mathbb{N}$, and $\gamma_0, \gamma_1, \dots \gamma_{n-1}$ be offsets in $[0,1[$. Assume that for any $p,q$ such that $0<q<p<n$ and any $r_0 \in \mathbb{Z}-\gamma_0$, $r_q \in \mathbb{Z}-\gamma_q$ and $r_p \in \mathbb{Z}-\gamma_p$ we have either Inequation (1) when $n$ is odd, or Inequation (2) when $n$ is even.*

$$(n \text{ odd}) \qquad r_0 \sin\tfrac{2(p-q)\pi}{n} + r_p \sin\tfrac{2q\pi}{n} - r_q \sin\tfrac{2p\pi}{n} \neq 0 \tag{1}$$

$$(n \text{ even}) \qquad r_0 \sin\tfrac{(p-q)\pi}{n} + r_p \sin\tfrac{q\pi}{n} - r_q \sin\tfrac{p\pi}{n} \neq 0 \tag{2}$$

*Then the grid $G_n(\gamma_0, \gamma_1, \dots \gamma_{n-1})$ is regular.*

**Proof.** We will prove this proposition by contradiction *i.e.* we assume a grid is singular and we show that it implies the existance of $r_0, r_p, r_q$ such that the Inequation (1) is contradicted if $n$ is odd, and Inequation (2) is contradicted if $n$ is even.

We will actually prove it for odd $n$, the proof for even $n$ is exactly the same and it is only needed to replace the formula of $\zeta_n^k$ which in the even case is $e^{\mathrm{i}\frac{k\pi}{n}}$ instead of $e^{\mathrm{i}\frac{2k\pi}{n}}$, which means that in the angles we remove the factor 2.

Let $n$ be an odd integer and let $\gamma_0, \gamma_1, \dots \gamma_{n-1} \in [0,1[$ such that $G_n(\gamma_0, \gamma_1, \dots \gamma_{n-1})$ is singular. This means that there exist $z \in \mathbb{C}$ at the intersection of three lines, up to relabeling and rotation we chose to consider it is at the intersection of $H(\zeta_n^0, \gamma_0)$, $H(\zeta_n^q, \gamma_q)$ and $H(\zeta_n^p, \gamma_p)$

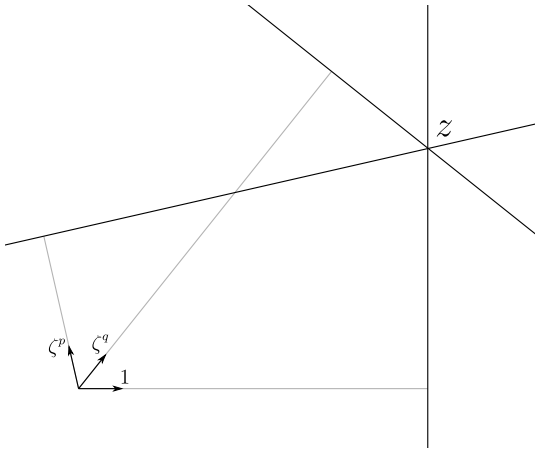



**Figure 5** Intersection of three lines.

for some $0 < q < p < n$, see Figure 5. This means that there exist $k_0, k_q, k_p \in \mathbb{Z}$ such that

$$\begin{cases} \mathrm{Re}(z) = k_0 - \gamma_0 \\ \mathrm{Re}(z \cdot \bar{\zeta_n^q}) = k_q - \gamma_q \\ \mathrm{Re}(z \cdot \bar{\zeta_n^p}) = k_p - \gamma_p \end{cases}$$

Write $z = k_0 - \gamma_0 + iy$ and $\zeta_n = e^{\frac{2i\pi}{n}}$. Now we have

$$\begin{cases} z = k_0 - \gamma_0 + iy \\ (k_0 - \gamma_0)\cos\frac{2q\pi}{n} + y\sin\frac{2q\pi}{n} = k_q - \gamma_q \\ (k_0 - \gamma_0)\cos\frac{2p\pi}{n} + y\sin\frac{2p\pi}{n} = k_p - \gamma_p \end{cases}$$

Let us now cancel out the $y$ terms by substituting the third line by $\sin\frac{2p\pi}{n}$ times the seconde equality minus $\sin\frac{2q\pi}{n}$ times the third equality.

$$\begin{cases} z = k_0 - \gamma_0 + iy \\ (k_0 - \gamma_0)\cos\frac{2q\pi}{n} + y\sin\frac{2q\pi}{n} = k_q - \gamma_q \\ (k_0 - \gamma_0)(\cos\frac{2q\pi}{n}\sin\frac{2p\pi}{n} - \cos\frac{2p\pi}{n}\sin\frac{2q\pi}{n}) = (k_q - \gamma_q)\sin\frac{2p\pi}{n} - (k_p - \gamma_p)\sin\frac{2q\pi}{n} \end{cases}$$

Now let us study the third line to simplify it.

$$\begin{aligned} &(k_0 - \gamma_0)\left(\cos\tfrac{2q\pi}{n}\sin\tfrac{2p\pi}{n} - \cos\tfrac{2p\pi}{n}\sin\tfrac{2q\pi}{n}\right) = (k_q - \gamma_q)\sin\tfrac{2p\pi}{n} - (k_p - \gamma_p)\sin\tfrac{2q\pi}{n} \\ &\Leftrightarrow (k_0 - \gamma_0)\sin\tfrac{2(p-q)\pi}{n} + (k_p - \gamma_p)\sin\tfrac{2q\pi}{n} - (k_q - \gamma_q)\sin\tfrac{2p\pi}{n} = 0 \end{aligned}$$

If we rewrite $k_0 - \gamma_0,\ k_p - \gamma_p,\ k_q - \gamma_q$ as $r_0,\ r_p,\ r_q$ we obtain

$$r_0\sin\tfrac{2(p-q)\pi}{n} + r_p\sin\tfrac{2q\pi}{n} - r_q\sin\tfrac{2p\pi}{n} = 0$$

which is exactly the contradiction of Inequation (1). ◀

In the next section we consider the solutions to these kind of trigonometric equations.

## 4 Trigonometric diophantine equations

We call *rational angles* the set $\pi\mathbb{Q}$. We consider now equations of the type

$$A\cos a + B\cos b + C\cos c = 0 \tag{3}$$

with $a, b$ and $c$ rational angles.

In the previous paragraph we had sine instead of cosine but we can always convert sine to cosine, and we had $A \in \mathbb{Z} - \gamma$ for some real number $0 \leqslant \gamma < 1$ and similarly for $B$ and $C$ but now we will consider $A, B$ and $C$ to be rationals.

The following result is from Conway and Jones.

▶ **Theorem 4** ([3]). *Suppose we have at most four distinct rational angles strictly between 0 and $\frac{\pi}{2}$ for which some rational linear combination of their cosines has rational value but no proper subset has this property.*
*Then the appropriate linear combination is proportional to one from the following list:*

$$\cos\tfrac{\pi}{3} = \tfrac{1}{2} \tag{4}$$
$$-\cos\phi + \cos\left(\tfrac{\pi}{3} - \phi\right) + \cos\left(\tfrac{\pi}{3} + \phi\right) = 0 \ \left(0 < \phi < \tfrac{\pi}{6}\right) \tag{5}$$
$$\cos\tfrac{\pi}{5} - \cos\tfrac{2\pi}{5} = \tfrac{1}{2} \tag{6}$$
$$\cos\tfrac{\pi}{7} - \cos\tfrac{2\pi}{7} + \cos\tfrac{3\pi}{7} = \tfrac{1}{2} \tag{7}$$
$$\cos\tfrac{\pi}{5} - \cos\tfrac{\pi}{15} + \cos\tfrac{4\pi}{15} = \tfrac{1}{2} \tag{8}$$
$$-\cos\tfrac{2\pi}{5} + \cos\tfrac{2\pi}{15} - \cos\tfrac{7\pi}{15} = \tfrac{1}{2} \tag{9}$$
$$\cos\tfrac{\pi}{7} + \cos\tfrac{3\pi}{7} - \cos\tfrac{\pi}{21} + \cos\tfrac{8\pi}{21} = \tfrac{1}{2} \tag{10}$$
$$\cos\tfrac{\pi}{7} - \cos\tfrac{2\pi}{7} + \cos\tfrac{2\pi}{21} - \cos\tfrac{5\pi}{21} = \tfrac{1}{2} \tag{11}$$
$$-\cos\tfrac{2\pi}{7} + \cos\tfrac{3\pi}{7} + \cos\tfrac{4\pi}{21} + \cos\tfrac{10\pi}{21} = \tfrac{1}{2} \tag{12}$$
$$-\cos\tfrac{\pi}{15} + \cos\tfrac{2\pi}{15} + \cos\tfrac{4\pi}{15} - \cos\tfrac{7\pi}{15} = \tfrac{1}{2} \tag{13}$$

See the original article [3] for the proof. The proof is based on a more general result on vanishing sums of roots of unity. And this is proved using complex numbers and the theory of vanishing formal sums. If we adapt this result for sums of three cosines that have value zero we get:

▶ **Corollary 5.** *Let $a \leqslant b \leqslant c$ be rational angles strictly between 0 and $\frac{\pi}{2}$ and not all equal, and let $A$, $B$, $C$ be non-zero rationals.*

*If $A\cos a + B\cos b + C\cos c = 0$ then either*

$$\begin{cases} a = \frac{\pi}{5} \\ b = \frac{\pi}{3} \\ c = \frac{2\pi}{5} \\ B = C = -A \end{cases} \quad \text{or} \quad \begin{cases} 0 < a < \frac{\pi}{6} \\ b = \frac{\pi}{3} - a \\ c = \frac{\pi}{3} + a \\ B = C = -A \end{cases}$$

**Proof.** We just need to apply Theorem 4. First remark that there is no solution for $A\cos a + B\cos b = 0$ with $a$ and $b$ distinct and strictly between 0 and $\frac{\pi}{2}$, and $A$ and $B$ non zero. Now with $0 < a < b < c < \frac{\pi}{2}$, we have either a combination of Equations (4) and (6) (first case) or Equation (5) (second case). ◀

## 5 Proof of Theorem 2

Here we use Proposition 3 and Corollary 5 to prove Theorem 2.

## 5.1 For odd $n$

First let us remark that in Theorem 2 the first statement when restricted to odd $n$ is a strict subcase of the second statement, so here we will prove the second statement which is as follows: for any odd $n \geqslant 3$ and any tuple of non-zero rational offsets $\gamma = (\gamma_i)_{0\leqslant i<n} \in (\mathbb{Q} \cap ]0,\, 1[)^n$ the $n$-fold multigrid $G_n(\gamma)$ is regular. We reformulate this with Proposition 3 as: for any odd $n \geqslant 3$, for any $0 < p < q < n$ and any three non-zero rational offsets $\gamma_0, \gamma_p, \gamma_q$, for any $r_0 \in \mathbb{Z} - \gamma_0$, $r_p \in \mathbb{Z} - \gamma_p$ and $r_q \in \mathbb{Z} - \gamma_q$ we have

$$r_0 \sin \tfrac{2(p-q)\pi}{n} + r_p \sin \tfrac{2q\pi}{n} - r_q \sin \tfrac{2p\pi}{n} \neq 0.$$

Actually we prove a slitghly reformulated version: for any odd $n \geqslant 3$, for any $0 < p < q < n$ and any three non-zero rationals $r_0, r_p, r_q \in \mathbb{Q}_{\setminus\{0\}}$ we have

$$r_0 \sin \tfrac{2(p-q)\pi}{n} + r_p \sin \tfrac{2q\pi}{n} - r_q \sin \tfrac{2p\pi}{n} \neq 0.$$

To apply Corollary 5 we first need to translate the formula with sine and with angles in $[0,\, 2\pi[$ as a formula with cosine and angles in $\left]0,\, \frac{\pi}{2}\right[$.

▶ **Lemma 6** (Sine and Cosine). *For $\theta \in [0,\, 2\pi[$ we have*

$$\sin\theta = (-1)^{\left\lfloor \frac{\theta}{\pi} \right\rfloor} \cos\left((-1)^{\left\lfloor \frac{2\theta}{\pi} \right\rfloor}(\left\lfloor \tfrac{\theta}{\pi} \right\rfloor \pi + \tfrac{\pi}{2} - \theta)\right)$$

*and $(-1)^{\left\lfloor \frac{2\theta}{\pi} \right\rfloor}(\left\lfloor \frac{\theta}{\pi} \right\rfloor \pi + \frac{\pi}{2} - \theta) \in \left[0,\, \frac{\pi}{2}\right]$.*

**Proof.** This result is just a rewriting of :

- if $0 \leqslant \theta < \frac{\pi}{2}$ then $\sin\theta = \cos(\frac{\pi}{2} - \theta)$ and $(\frac{\pi}{2} - \theta) \in \left[0,\, \frac{\pi}{2}\right]$
- if $\frac{\pi}{2} \leqslant \theta < \pi$ then $\sin\theta = \cos(\theta - \frac{\pi}{2})$ and $(\theta - \frac{\pi}{2}) \in \left[0,\, \frac{\pi}{2}\right]$
- if $\pi \leqslant \theta < \frac{3\pi}{2}$ then $\sin\theta = -\cos(\frac{3\pi}{2} - \theta)$ and $(\frac{3\pi}{2} - \theta) \in \left[0,\, \frac{\pi}{2}\right]$
- if $\frac{3\pi}{2} \leqslant \theta < 2\pi$ then $\sin\theta = -\cos(\theta - \frac{3\pi}{2})$ and $(\theta - \frac{3\pi}{2}) \in \left[0,\, \frac{\pi}{2}\right]$ ◀

We define $\epsilon(\theta) := (-1)^{\left\lfloor \frac{\theta}{\pi} \right\rfloor}$ and $\phi(\theta) := (-1)^{\left\lfloor \frac{2\theta}{\pi} \right\rfloor}(\left\lfloor \frac{\theta}{\pi} \right\rfloor \pi + \frac{\pi}{2} - \theta)$. Remark that this means that $\theta = \left\lfloor \frac{\theta}{\pi} \right\rfloor \pi + \frac{\pi}{2} - (-1)^{\left\lfloor \frac{2\theta}{\pi} \right\rfloor}\phi(\theta)$.

Let $n, p, q$ be integers such that $n$ is odd, $n \geqslant 3$ and $0 < q < p < n$. By contradiction suppose that there exists $r_0$,$r_p$ and $r_q$ non-zero rationals such that

$$r_0 \sin \tfrac{2(p-q)\pi}{n} + r_p \sin \tfrac{2q\pi}{n} - r_q \sin \tfrac{2p\pi}{n} = 0.$$

By Lemma 6 we have

$$r_0\epsilon(\tfrac{2(p-q)\pi}{n}) \cos(\phi(\tfrac{2(p-q)\pi}{n})) + r_p\epsilon(\tfrac{2q\pi}{n}) \cos(\phi(\tfrac{2q\pi}{n})) - r_q\epsilon(\tfrac{2p\pi}{n}) \cos(\phi(\tfrac{2p\pi}{n})) = 0.$$

Which we reformulate as

$$r'_0 \cos\theta_0 + r'_p \cos\theta_q + r'_q \cos\theta_p = 0,$$

with $r'_0 := r_0\epsilon(\frac{2(p-q)\pi}{n})$, $r'_p := r_p\epsilon(\frac{2q\pi}{n})$, $r'_q := -r_q\epsilon(\frac{2p\pi}{n})$ and $\theta_0 := \phi(\frac{2(p-q)\pi}{n})$, $\theta_q := \phi(\frac{2q\pi}{n})$, $\theta_p := \phi(\frac{2p\pi}{n})$.

Remark that for odd $n$ and any $0 < k < n$ we have $\frac{2k\pi}{n} \notin \{0, \frac{\pi}{2}, \pi, \frac{3\pi}{2}\}$, so $\phi(\frac{2k\pi}{n}) \notin \{0, \frac{\pi}{2}\}$. This implies that $0 < \theta_0, \theta_p, \theta_q < \frac{\pi}{2}$.

Moreover for odd $n$ we have that $\theta_0, \theta_p, \theta_q$ are not all equal. By contradiction if $\theta_0 = \theta_p = \theta_q$ we have that $\frac{2p\pi}{n}, \frac{2q\pi}{n}, \frac{2(p-q)\pi}{n} \in \phi^{-1}(\{\theta_0\}) = \{\frac{\pi}{2} - \theta_0, \frac{\pi}{2} + \theta_0, \frac{3\pi}{2} - \theta_0, \frac{3\pi}{2} + \theta_0\}$ which is impossible. So we have

$$r'_0 \cos\theta_0 + r'_p \cos\theta_q + r'_q \cos\theta_p = 0,$$

with non-zero rationals $r'_0, r'_p, r'_q$ and with three angles strictly between 0 and $\frac{\pi}{2}$ and not all equal.

So we can apply Corollary 5 and we now have two cases:

1. $\{\theta_0, \theta_p, \theta_q\} = \{\frac{\pi}{5}, \frac{\pi}{3}, \frac{2\pi}{5}\}$
2. $\{\theta_0, \theta_p, \theta_q\} = \{\theta, \frac{\pi}{3} - \theta, \frac{\pi}{3} + \theta\}$ for some $0 < \theta < \frac{\pi}{6}$

Let us now show that both cases lead to a contradiction. In the first case we have that $\{\frac{2p\pi}{n}, \frac{2q\pi}{n}, \frac{2(p-q)\pi}{n}\} = \{\theta_1, \theta_2, \theta_3\}$ with

$$\theta_1 \in \phi^{-1}(\{\tfrac{\pi}{5}\}) = \{\tfrac{3\pi}{10}, \tfrac{7\pi}{10}, \tfrac{13\pi}{10}, \tfrac{17\pi}{10}\}$$
$$\theta_2 \in \phi^{-1}(\{\tfrac{\pi}{3}\}) = \{\tfrac{\pi}{6}, \tfrac{5\pi}{6}, \tfrac{7\pi}{6}, \tfrac{11\pi}{6}\}$$
$$\theta_3 \in \phi^{-1}(\{\tfrac{2\pi}{5}\}) = \{\tfrac{\pi}{10}, \tfrac{9\pi}{10}, \tfrac{11\pi}{10}, \tfrac{19\pi}{10}\}$$

This is impossible because by definition $\frac{2p\pi}{n} = \frac{2q\pi}{n} + \frac{2(p-q)\pi}{n}$ and we have no $\theta_1, \theta_2, \theta_3$ as defined above such that one is the sum of the two other.

In the second case we have that $\{\theta_0, \theta_p, \theta_q\} = \{\theta, \frac{\pi}{3} - \theta, \frac{\pi}{3} + \theta\}$ for some $0 < \theta < \frac{\pi}{6}$. Now we use $\frac{2p\pi}{n} = \frac{2q\pi}{n} + \frac{2(p-q)\pi}{n}$ and by definition we have

$$\frac{2p\pi}{n} = \left\lfloor \tfrac{2p}{n} \right\rfloor \pi + \tfrac{\pi}{2} - (-1)^{\left\lfloor \frac{4p}{n} \right\rfloor}\theta_p = \left\lfloor \tfrac{2p}{n} \right\rfloor \pi + \tfrac{\pi}{2} \pm \theta_p$$
$$\frac{2q\pi}{n} = \left\lfloor \tfrac{2q}{n} \right\rfloor \pi + \tfrac{\pi}{2} - (-1)^{\left\lfloor \frac{4q}{n} \right\rfloor}\theta_q = \left\lfloor \tfrac{2q}{n} \right\rfloor \pi + \tfrac{\pi}{2} \pm \theta_q$$
$$\frac{2(p-q)\pi}{n} = \left\lfloor \frac{2(p-q)}{n} \right\rfloor \pi + \tfrac{\pi}{2} - (-1)^{\left\lfloor \frac{4(p-q)}{n} \right\rfloor}\theta_0 = \left\lfloor \frac{2(p-q)}{n} \right\rfloor \pi + \tfrac{\pi}{2} \pm \theta_0$$

By assembling these two we get

$$(\left\lfloor \tfrac{2p}{n} \right\rfloor - \left\lfloor \tfrac{2q}{n} \right\rfloor - \left\lfloor \frac{2(p-q)}{n} \right\rfloor - 1)\pi + \tfrac{\pi}{2} = \pm\theta_p \pm \theta_0 \pm \theta_q$$

And with $\{\theta_0, \theta_p, \theta_q\} = \{\theta, \frac{\pi}{3} - \theta, \frac{\pi}{3} - \theta\}$ we get

$$(\pm\theta_p \pm \theta_0 \pm \theta_q) \in \{\pm 3\theta, \pm\theta, \pm\tfrac{2\pi}{3} \pm \theta\}$$

However this is impossible since for $0 < \theta < \frac{\pi}{6}$, we have

$$\left(\mathbb{Z}\pi + \tfrac{\pi}{2}\right) \cap \{\pm 3\theta, \pm\theta, \pm\tfrac{2\pi}{3} \pm \theta\} = \emptyset$$

By contradiction we proved that for odd $n$, any $n$-fold multigrid with non-zero rational offsets is regular.

## 5.2 For even $n$

Let us now prove the first statement of Theorem 2 for even $n$ which is: for any even $n \geqslant 4$ and any non-zero rational offset $r \in \mathbb{Q} \cap ]0, 1[$ the $n$-fold multigrid $G_n(r)$ is regular. We reformulate it using Proposition 3 as: for any even $n \geqslant 4$ and any non-zero rational offset $r \in \mathbb{Q} \cap ]0, 1[$, for any $0 < q < p < n$ and any $r_0 \in \mathbb{Z} - r$, $r_p \in \mathbb{Z} - r$ and $r_q \in \mathbb{Z} - r$ we have

$$r_0 \sin \tfrac{(p-q)\pi}{n} + r_p \sin \tfrac{q\pi}{n} - r_q \sin \tfrac{p\pi}{n} \neq 0.$$

We will prove this by contradiction. Let $n \geqslant 4$ be an even integer and $r$ be a non-zero rational offset. Suppose that there exists $p, q, r_0, r_p, r_q$ with $p, q$ integers, $0 < q < p < n$ and $r_0 \in \mathbb{Z} - r$, $r_p \in \mathbb{Z} - r$ and $r_q \in \mathbb{Z} - r$, such that

$$r_0 \sin \tfrac{(p-q)\pi}{n} + r_p \sin \tfrac{q\pi}{n} - r_q \sin \tfrac{p\pi}{n} = 0.$$

We apply Lemma 6 with the fact that since $0 < \frac{p\pi}{n}, \frac{q\pi}{n}, \frac{(p-q)\pi}{n} < \pi$ we have $\epsilon(\frac{k\pi}{n}) = 1$ and $\phi(\frac{k\pi}{n}) = (-1)^{\left\lfloor \frac{2k}{n} \right\rfloor}(\frac{\pi}{2} - \theta)$ for $k \in \{p, q, (p-q)\}$. We obtain

$$r_0 \cos\theta_0 + r_p \cos\theta_q - r_q \cos\theta_p = 0$$

with $\theta_0 = \phi(\frac{(p-q)\pi}{n})$, $\theta_p = \phi(\frac{p\pi}{n})$ and $\theta_q = \phi(\frac{q\pi}{n})$. And since $0 < \frac{p\pi}{n}, \frac{q\pi}{n}, \frac{(p-q)\pi}{n} < \pi$ we have $\theta_0, \theta_p, \theta_q \in \left[0, \frac{\pi}{2}\right[$. In particular since $n$ is even we can have $\frac{p\pi}{n} = \frac{\pi}{2}$ which means that we can have $\theta_p = 0$ (and also for $\theta_0$ or $\theta_q$). Note also that with $n$ even (contrary to the odd case) we can have $\theta_0 = \theta_p = \theta_q$, for example with $n = 6$, $q = 2$ and $p = 4$ we have $\theta_0 = \theta_p = \theta_q = \frac{\pi}{6}$. Which means that now we have a disjunction of four cases:

1. $\theta_0 = \theta_p = \theta_q$
2. $0 < \theta_0, \theta_p, \theta_q < \frac{\pi}{2}$ and not all equal
3. two of the angles are 0 and the other one is not
4. one of the angles is 0 and the other two are not

The first case reduces to $(r_0 + r_p - r_q)\cos\theta_0 = 0$ and with $\theta_0 \in \left[0, \frac{\pi}{2}\right[$ we have $\cos\theta_0 \neq 0$ so $(r_0 + r_p - r_q) = 0$ but this is impossible because $(r_0 + r_p - r_q) \in \mathbb{Z} - r$ and $0 \notin (\mathbb{Z} - r)$ for $r \in (\mathbb{Q} \cap ]0, 1[)$.

The second case is the same as the one discussed in Subsection 5.1 above, the main thing we used in the proof for odd $n$ is the fact that $\frac{2p\pi}{n} = \frac{2q\pi}{n} + \frac{2(p-q)\pi}{n}$ but we have the same for even $n$ with $\frac{p\pi}{n} = \frac{q\pi}{n} + \frac{(p-q)\pi}{n}$. So the proof holds and this case is impossible.

The third case is impossible because for two angles to be 0, we need two of $\{\frac{p\pi}{n}, \frac{q\pi}{n}, \frac{(p-q)\pi}{n}\}$ to be $\frac{\pi}{2}$ but this is impossible with $0 < q < p < n$.

The fourth case reduces to $A\cos a + B\cos b = C$ with $A, B, C \in \mathbb{Z} - r \subset \mathbb{Q}$ so we can apply Theorem 4 and we get that either $a = b = \frac{\pi}{3}$ or $a = \frac{\pi}{5}$ and $b = \frac{2\pi}{5}$. The first subcase is impossible because $\{\theta_0, \theta_p, \theta_q\} = \{0, \frac{\pi}{3}\}$ implies $\{\frac{p\pi}{n}, \frac{q\pi}{n}, \frac{(p-q)\pi}{n}\} \subseteq \{\frac{\pi}{6}, \frac{\pi}{2}, \frac{5\pi}{6}\}$ and this is incompatible with $\frac{p\pi}{n} = \frac{q\pi}{n} + \frac{(p-q)\pi}{n}$. The second subcase is also impossible for the same reason as (up to interchanging $\theta_0$, $\theta_p$ and $\theta_q$) we would have $\theta_0 = 0$, $\theta_p = \frac{\pi}{5}$ and $\theta_q = \frac{2\pi}{5}$ which means that $\frac{(p-q)\pi}{n} = \frac{\pi}{2}$, $\frac{p\pi}{n} \in \{\frac{3\pi}{10}, \frac{7\pi}{10}\}$ and $\frac{q\pi}{n} \in \{\frac{\pi}{10}, \frac{9\pi}{10}\}$ and this is incompatible with $\frac{p\pi}{n} = \frac{q\pi}{n} + \frac{(p-q)\pi}{n}$.

Note that only in the first case we use the fact that the offset are all the same $r \in \mathbb{Q} \cap ]0, 1[$, the three other case work for any non-zero rational offsets as for the odd cases. And actually we could refine the condition because what is important here is that $0 \notin (\mathbb{Z} - \gamma_0 - \gamma_q + \gamma_p)$ with $\gamma_0$, $\gamma_p$ and $\gamma_q$ the rational offsets. So for even $n$ if we have a tuple of rational offsets $\gamma = (\gamma_i)_{0 \leqslant i < n}$ such that for any distinct $i$, $j$, $k$ we have $\gamma_i - \gamma_j - \gamma_k \neq 0$ then the multigrid $G_n(\gamma)$ is regular.

## References


1 M. Baake and U. Grimm. *Aperiodic Order: A Mathematical Invitation*, volume 1. Cambridge University Press, 2013. doi:10.1017/9781139033862.

2 F.P.M. Beenker. Algebraic theory of non-periodic tilings of the plane by two simple building blocks : a square and a rhombus, 1982. Technical report.

3 J. Conway and A. Jones. Trigonometric diophantine equations (on vanishing sums of roots of unity). *Acta Arithmetica*, 30(3):229–240, 1976. doi:10.4064/aa-30-3-229-240.

4 N. G. De Bruijn. Algebraic theory of penrose's nonperiodic tilings of the plane. i and ii. *Kon. Nederl. Akad. Wetensch. Proc. Ser. A*, 43(84):39–66, 1981. doi:10.1016/1385-7258(81)90016-0.

5 F. Gähler and J. Rhyner. Equivalence of the generalised grid and projection methods for the construction of quasiperiodic tilings. *Journal of Physics A: Mathematical and General*, 19(2):267, 1986. doi:10.1088/0305-4470/19/2/020.

6 J. Kari and V. H. Lutfalla. Substitution planar tilings with $n$-fold rotational symmetry, 2020. arXiv:2010.01879.

7 V. H. Lutfalla. n-fold multigrid dual tilings, 2021. Software repository. doi:10.5281/zenodo.4698387.

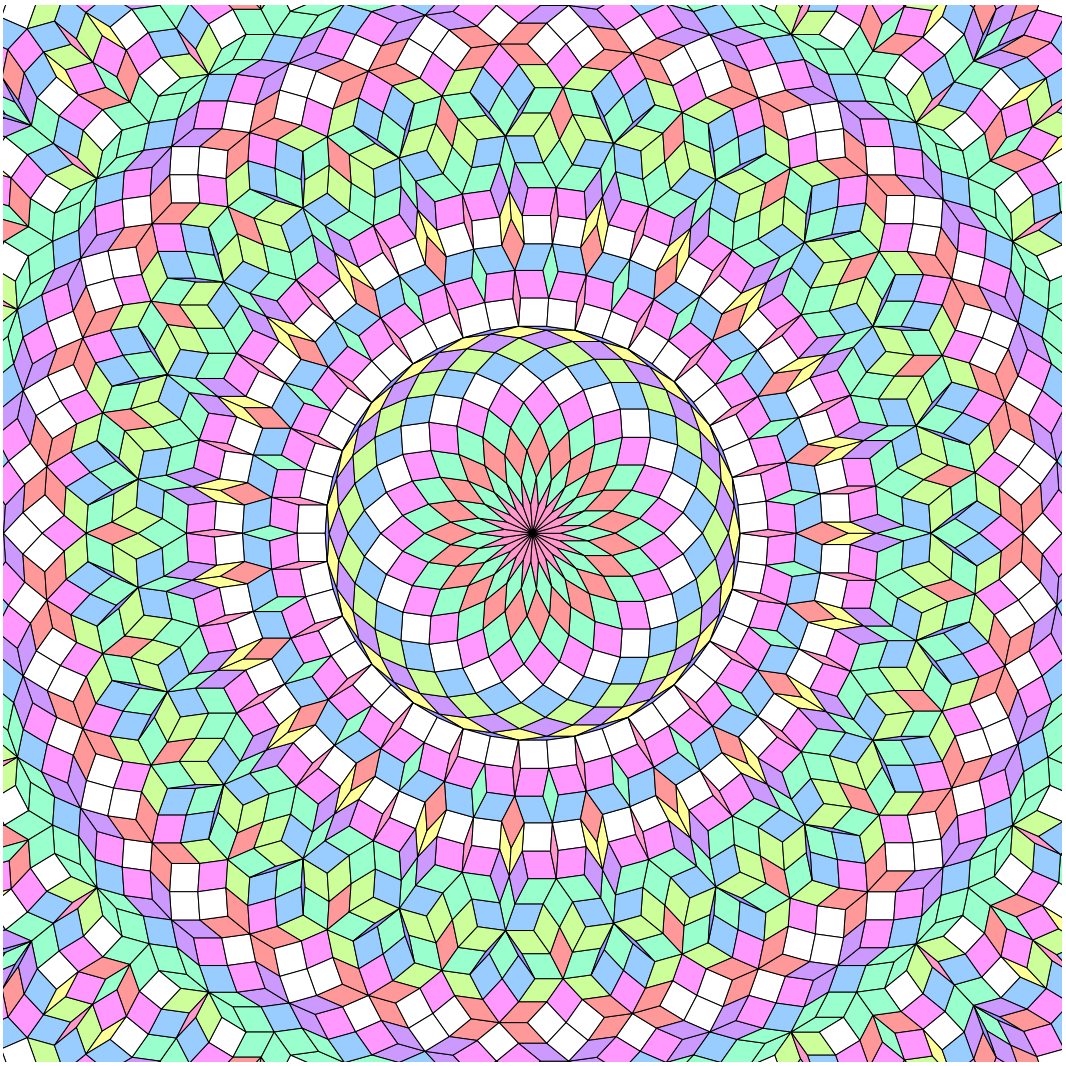


**Figure 6** A central patch of the multigrid dual tiling $P_{23}(\frac{1}{23})$ with 23-fold rotational symmetry.


8 R. Penrose. The role of aesthetics in pure and applied mathematical research. *Bull. Inst. Math. Appl.*, 10:266–271, 1974.

9 M. Senechal. *Quasicrystals and geometry.* CUP Archive, 1996.

# Non-Deterministic Updates of Boolean Networks

**Loïc Paulevé**
Université Bordeaux, Bordeaux INP, CNRS, LaBRI, UMR5800, F-33400 Talence, France

**Sylvain Sené**
Université Publique, Marseille, France

## Abstract

Boolean networks are discrete dynamical systems where each automaton has its own Boolean function for computing its state according to the configuration of the network. The updating mode then determines how the configuration of the network evolves over time. Many of updating modes from the literature, including synchronous and asynchronous modes, can be defined as the composition of elementary deterministic configuration updates, i.e., by functions mapping configurations of the network. Nevertheless, alternative dynamics have been introduced using ad-hoc auxiliary objects, such as that resulting from binary projections of Memory Boolean networks, or that resulting from additional pseudo-states for Most Permissive Boolean networks. One may wonder whether these latter dynamics can still be classified as updating modes of finite Boolean networks, or belong to a different class of dynamical systems. In this paper, we study the extension of updating modes to the composition of non-deterministic updates, i.e., mapping sets of finite configurations. We show that the above dynamics can be expressed in this framework, enabling a better understanding of them as updating modes of Boolean networks. More generally, we argue that non-deterministic updates pave the way to a unifying framework for expressing complex updating modes, some of them enabling transitions that cannot be computed with elementary and non-elementary deterministic updates.

**2012 ACM Subject Classification** Theory of computation → Models of computation; Theory of computation → Program semantics; Applied computing → Systems biology

**Keywords and phrases** Natural computing, discrete dynamical systems, semantics

**Digital Object Identifier** 10.4230/OASIcs.AUTOMATA.2021.10

**Funding** *Loïc Paulevé*: French Agence Nationale pour la Recherche (ANR): ANR-FNR project "AlgoReCell" ANR-16-CE12-0034 and ANR project "BNeDiction" ANR-20-CE45-0001.
*Sylvain Sené*: his salary as a French state agent affiliated to Université Aix-Marseille, Université Toulon, CNRS, LIS, UMR 7020, Marseille, France, and the ANR project "FANs" ANR-18-CE40-0002.

## 1 Introduction

Boolean networks (BNs) are formal dynamical systems composed of automata, each of them having a Boolean state. A major difference between BNs and cellular automata (CAs) is that each automaton of a BN follows its own rules for computing its next state depending on the states of the other automata in the network. Consequently, whereas influences between cells in a CA are structured homogeneously according to a cellular space, those between automata in a BN are structured according to any directed graph. In this paper, only finite BNs are considered, as it is generally the case in the literature, notably because BNs are mostly viewed as both a real-world computational model and a real-world modeling framework.

The study of BNs led to fundamental results linking the network architecture (structure of influences between automata) to the existence of fixed points and to the number of limit cycles they can exhibit [1, 10, 4]. Notably, it is well known that such limit behaviors may depend on the way automata update their state over time [3, 12, 2, 22]. This emphasizes the importance of what is classically called the updating modes in the analyses of BNs.


BNs are widely employed to model natural systems, with prominent applications in biology. These applications inspired the definition of various updating modes aiming at reflecting constraints related to the quantitative nature of the abstracted system, such as reaction duration and influence thresholds. There is actually no consensus about one updating mode that would be the most likely, the most representative of the biological reality. As a consequence, the choice of this or that updating mode strongly depends on the problematics, on the nature of the questions addressed. Thus, it remains essential to analyze the impact of a wide range of updating modes with distinct features.

In this paper, we address the formalization of updating modes in the framework of BNs. From a very general perspective, given a BN and one of its configurations, an updating mode specify how to compute the possible next configurations (plural implying non-deterministic systems).

A large majority of updating modes introduced so far can be expressed using deterministic functions mapping the configurations of the network. This leads to *elementary* transitions, as it is the case with synchronous (or parallel) and asynchronous [23] updating modes, which may result in non-deterministic dynamics. These functions may also be composed, as in block-sequential [25] and block-parallel [11] updating modes, generating non-elementary transitions.

These compositions of deterministic *updates*, however, do not cover all the updating modes introduced in the literature. Indeed, updating modes may also make use of parameters that cannot *a priori* and intuitively be directly captured by these deterministic updates. These parameters can represent kinds of delays or threshold effects of state changes. In this paper, we focus on 3 examples of BN dynamics which have been recently introduced and defined using ad-hoc formalizations:

- *Memory Boolean networks* (MBNs) [14, 15] take into account some kind of delay for the decrease of automata. They have been introduced by the means of a deterministic dynamical system with non-binary configurations, whose updates are computed deterministically from the BN and a memory vector, specifying the delay for each automaton.
- *Interval Boolean networks* (IBNs) [7] account for a duration for updating an automaton. The other automata can be updated until the former automaton eventually change of state. They have been defined by an encoding as the fully-asynchronous updating of a BN of dimension $2n$. The dynamics of the original BN are then recovered by projection.
- *Most Permissive Boolean networks* (MPBNs) [24] bring a formal abstraction of trajectories of quantitative models which are compatible with the BN formalism: from an initial configuration, if there is no trajectory where a given automaton is 1 (or 0), then, no quantitative refinement of the model can increase (or decrease) the value of this automaton. MPBNs have been defined by introducing additional states for automata to account for their state change (increasing and decreasing). An automaton in one of these states can be read non-deterministically as 0 or 1.

Overall, the definition of these BN dynamics involve either non-Boolean configurations, projections of higher-dimension BN, or both. Importantly, they suggest that deterministic updates are not expressive enough to capture specific dynamics. This is striking with IBNs and MPBNs which can generate transitions that are neither elementary nor non-elementary transitions, and thus predict trajectories that are impossible with the asynchronous updating mode.

We show that these dynamics can all be expressed using Boolean configurations in a simple generic framework, which extends the deterministic updates to *non-deterministic* updates: functions mapping sets of configurations. In the case of MBNs, the obtained

definition from the binary projections of their deterministic discrete dynamics actually help to understand the generated dynamics: the transitions match with a particular subset of elementary transitions, suggesting a simpler parameterization. In the case of IBNs and MPBNs, the transitions extend the elementary and non-elementary transitions by considering some delay for the state changes, and having different interpretation of how to "read" an automaton in the course of state change. The obtained definitions suggest many variants for generating sub-dynamics, similarly to the asynchronous mode which generates all elementary transitions.

Thus, non-deterministic updates offer a unified yet simple framework for defining and understanding BN updating modes with more expressivity than usual deterministic updates. However, should any set update be considered as a BN updating mode? We propose an argumentation for a *reasonable* updating mode in the last section, where we suggest that the state change should always be justified by the application of a local function. This suggests that the MP updating mode generates the largest set of transitions that fulfill this criterion.

**Notations.** The Boolean domain $\{0,1\}$ is denoted by $\mathbb{B}$; the set $\{1,\cdots,n\}$ is denoted by $[\![n]\!]$. Given a finite domain $A$ with a partial order $\preceq$, and a function $h$ mapping elements of $A$ to $A$, for any $k \in \mathbb{N}_{>0}$, we write $h^k$ for $h$ iterated $k$ times. Whenever for any $a \in A$, $a \preceq h(a)$, we write $h^\omega$ for the iteration of $h$ until reaching a fixed point (in this paper, $A$ is often a power set with $\preceq$ being the subset relation).

## 2 Boolean networks and dynamics

A *Boolean network* (BN) of dimension $n$ is specified by a function $f : \mathbb{B}^n \to \mathbb{B}^n$ mapping Boolean vectors of dimension $n$. The components $[\![n]\!]$ of the BN are called *automata*. For each automaton $i \in [\![n]\!]$, $f_i : \mathbb{B}^n \to \mathbb{B}$ is the $i$-th component of this function, that we call the *local function* of automaton $i$. The $2^n$ Boolean vectors of $\mathbb{B}^n$ are called the *configurations* of the BN. In a configuration $x \in \mathbb{B}^n$, $x_i$ is the *state* of automaton $i$.

**Updating modes.** Given a BN $f$ of dimension $n$ and one of its configurations $x \in \mathbb{B}^n$, an *updating mode* $\mu$ characterizes the possible evolutions of $x$ with respect to $f(x)$. The dynamical system $(f,\mu)$ defines a binary *transition relation* between configurations of $\mathbb{B}^n$ denoted by $\to_{(f,\mu)} \subseteq \mathbb{B}^n \times \mathbb{B}^n$. This dynamical system can be represented by a directed graph $\mathcal{D}_{(f,\mu)} = (\mathbb{B}^n, \to_{(f,\mu)})$. This graph is usually called the *transition graph* of $(f,\mu)$. The reflexive and transitive closure of relation $\to_{(f,\mu)}$, denoted by $\to^*_{(f,\mu)}$ can be defined as follows: given two configurations $x, y \in \mathbb{B}^n$, $x \to^*_{(f,\mu)} y$ if and only if $x = y$ or there exists a path from $x$ to $y$ in $\mathcal{D}_{(f,\mu)}$.

A *deterministic* updating mode ensures that, for any BN $f$ of dimension $n$, each configuration has at most one outgoing transition ($\forall x, y, z \in \mathbb{B}^n$, $x \to_{(f,\mu)} y$ and $x \to_{(f,\mu)} z$ only if $y = z$). Otherwise, the updating mode is qualified as *non-deterministic*.

In the following, we consider the BN $f$ to be fixed, and thus, for the sake of simplicity, we omit the subscript $f$: the transition relation is denoted by $\to_\mu$ and the transition graph by $\mathcal{D}_\mu$.

**Dynamical properties.** A configuration $x \in \mathbb{B}^n$ is *transient* if there exists a configuration $y$ such that $x \to^*_\mu y$ and $y \not\to^*_\mu x$. Configurations that are not transient are called *limit configurations*. Because $n$ is finite, these configurations induce the terminal strongly connected components of $\mathcal{D}_\mu$, called the *limit sets* of $(f,\mu)$. If there exists at least one path from a

**Table 1** Configurations, local functions ($(f_i)_{i\in[\![3]\!]}$) and four updating functions ($\phi_\emptyset$, $\phi_1$, $\phi_{\{2,3\}}$, and $\phi_{[\![3]\!]}$) of Boolean network $f$ presented in Example 2.

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

transient configuration to a limit set, this limit set is called an *attractor* of $(f,\mu)$ [8, 21]. The *basin of attraction* of an attractor $\mathcal{A}$ of $(f,\mu)$, denoted by $\mathcal{B}(\mathcal{A})$, is the sub-graph of $\mathcal{D}_\mu$ induced by the set of transient configurations $x$ such that, for any limit configuration $y$ belonging to $\mathcal{A}$, $x \to_\mu^* y$. A limit set of cardinal 1, *i.e.* composed of a unique limit configuration $x$ is called a *fixed point* of $(f,\mu)$. A limit set of cardinal greater than 1 is called a *limit cycle* of $(f,\mu)$.

## 3 Updating modes with deterministic updates

### Elementary transitions

Let us consider a BN $f$ of dimension $n$ and one of its configurations $x\in\mathbb{B}^n$. Whenever $x$ and $f(x)$ differ by more than one component, one may define several ways to update $x$: either by replacing it with $f(x)$, i.e., applying simultaneously the local functions on every automata, or by modifying the state of only a subset of automata. For each set of automata to update, we obtain a deterministic function mapping configurations, that we refer to as an *elementary* deterministic update:

▶ **Definition 1.** *Given a BN $f$ of dimension $n$ and a set of automata $W\subseteq[\![n]\!]$, $\phi_W:\mathbb{B}^n\to\mathbb{B}^n$ is an* elementary deterministic update *with*

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

Whenever referring to singleton sets $\{i\}$ with $i\in[\![n]\!]$, we write $\phi_i$ instead of $\phi_{\{i\}}$. Notice that $\phi_{[\![n]\!]}=f$.

▶ **Example 2.** Let us consider the BN $f$ of dimension $n=3$ with $f(x)=\begin{pmatrix} f_1(x)=\neg x_3\\ f_2(x)=\neg x_1\wedge x_3\\ f_3(x)=\neg x_1\end{pmatrix}$.

Table 1 shows four distinct updatings on its configurations. The first updating is ineffective and consists in changing nothing. The second updating changes the state of automaton 1 by application of $\phi_1$, the third one changes the states of both automata 2 and 3 by application of $\phi_{\{2,3\}}$, and the fourth one changes the state of every automaton by application of $\phi_{[\![3]\!]}$.

We can then define the notion of *elementary transitions* of a BN, that are the transitions obtained by applying any elementary update on a non-empty subset of automata.

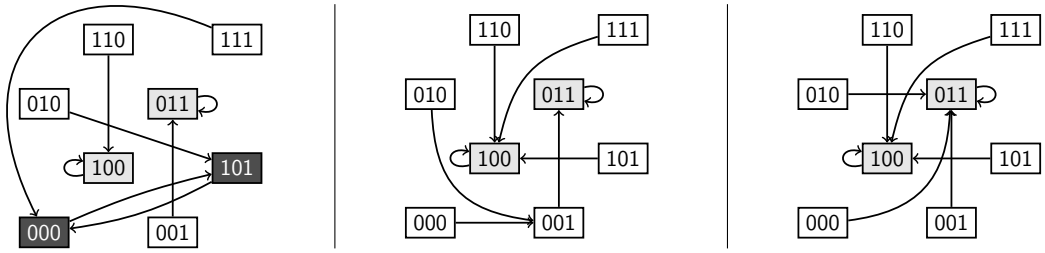



**Figure 1** Distinct possible block-sequential dynamics of BN $f$ defined in Example 2: (left panel) its parallel dynamics associated with ordered partition ($[\![3]\!]$); (central panel) the block-sequential dynamics associated with $(\{2,3\},\{1\})$; (right panel) the sequential dynamics associated with $(\{3\},\{1\},\{2\})$.

▶ **Definition 3.** *Given a BN $f$, its* elementary *transitions $\to_{\mathsf{e}} \subseteq \mathbb{B}^n \times \mathbb{B}^n$ are such that, for all configurations $x, y \in \mathbb{B}^n$, $x \to_{\mathsf{e}} y$ if and only if there exists a non-empty subset of automata $W \subseteq [\![n]\!]$ with $y = \phi_W(x)$.*

Let us now define some classical deterministic and non-deterministic updating modes from these elementary updates.

### Examples of deterministic updating modes

The most direct updating mode is the application of $f$ to the configuration $x$, resulting in the configuration $f(x)$, or, equivalently, $\phi_{[\![n]\!]}(x)$:

▶ **Definition 4.** *The* synchronous *(or* parallel*) updating mode of a BN $f$ of dimension $n$ generates the transition relation $\to_{\mathsf{p}} \subseteq \mathbb{B}^n \times \mathbb{B}^n$ such that, for all configurations $x, y \in \mathbb{B}^n$, $x \to_{\mathsf{p}} y$ if and only if $y = \phi_{[\![n]\!]}(x)$.*

*Sequential* updating modes are parameterized by a permutation of $[\![n]\!]$, fixing an ordering of elementary updates of single automata [13, 17, 9]. They can be generalized to *block-sequential* updating modes [25, 3, 16], parameterized by a permutation of a partition of $[\![n]\!]$:

▶ **Definition 5.** *Given a BN $f$ of dimension $n$ and $\mathsf{bs} = (W_1, \cdots, W_p)$ an ordered partition of $[\![n]\!]$, the* block-sequential *updating mode generates the transition relation $\to_{\mathsf{bs}} \subseteq \mathbb{B}^n \times \mathbb{B}^n$ such that, for all configurations $x, y \in \mathbb{B}^n$, $x \to_{\mathsf{bs}} y$ if and only if $y = \phi_{W_p} \circ \cdots \phi_{W_1}(x)$.*

Remark that the transitions of sequential and block-sequential modes may not be elementary. However, they always correspond to a path of elementary transitions: $x \to_{\mathsf{bs}} y$ only if $x \to_{\mathsf{e}}^* y$.

Going further in generalization, one may consider deterministic updating modes as infinite sequences of sets of automata, so that automata of a same subset execute their local function in parallel while the subsets are iterated sequentially. Remark that any of these possible deterministic updating modes will generate transitions corresponding to specific paths of elementary transitions.

### Examples of non-deterministic updating modes

It is important to notice that deterministic updates can lead to non-deterministic dynamics by allowing different updates on a same configuration. The most obvious example is the *asynchronous* mode[1] consisting of all the elementary transitions.

[1] The *asynchronous* mode is often referred to as *general asynchronous* in the systems biology modeling community.

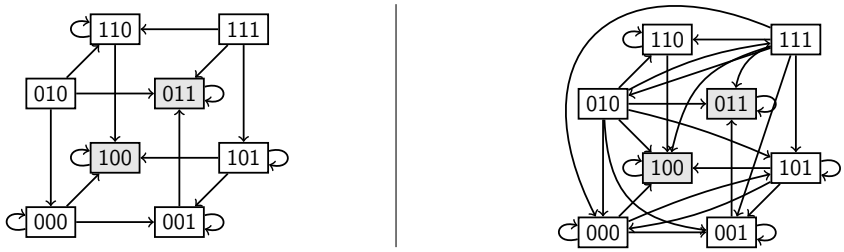



**Figure 2** Fully-asynchronous (left) and asynchronous (right) dynamics of BN $f$ defined in Example 2.

▶ **Definition 6.** *The* asynchronous *updating mode of a BN $f$ generates the transition relation* $\to_{\mathsf{a}} \subseteq \mathbb{B}^n \times \mathbb{B}^n$ *as* $\to_{\mathsf{a}} = \to_{\mathsf{e}}$.

One of the most usual non-deterministic updating modes of BNs is the *fully-asynchronous* mode[2], where only one automaton is updated in a transition. It is largely employed for the analysis of models of biological systems, arguing it enables capturing (some) behaviors caused by different time scale for automata updates.

▶ **Definition 7.** *The* fully-asynchronous *updating mode of a BN $f$ generates the transition relation* $\to_{\mathsf{fa}} \subseteq \mathbb{B}^n \times \mathbb{B}^n$ *such that, for all configurations* $x, y \in \mathbb{B}^n$*:* $x \to_{\mathsf{fa}} y$ *if and only if there exists* $i \in [\![n]\!]$ *with* $y = \phi_i(x)$.

Figure 2 shows the dynamics generated by the fully-asynchronous and asynchronous updating modes on the BN of Example 2.

## 4 Non-deterministic updates as set updates

The updates considered so far are deterministic, and can thus be defined as functions mapping configurations, i.e., of the form $\phi : \mathbb{B}^n \to \mathbb{B}^n$. As we have seen above, deterministic updates can generate non-deterministic updating modes, by allowing different updates to be applied on a same configuration.

Let us now extend to non-deterministic updates, that we model by functions mapping *sets of configurations*, i.e., of the form $\Phi : 2^{\mathbb{B}^n} \to 2^{\mathbb{B}^n}$. We define $\Phi$ as a map from sets of configurations to sets of configurations for enabling iterations and compositions of non-deterministic updates. Nevertheless, we assume that for any $X \subseteq \mathbb{B}^n$, $\Phi(X) = \bigcup_{x \in X} \Phi(\{x\})$: one can define $\Phi$ only from all singleton configuration set. This restriction ensures that, for any $X \subseteq \mathbb{B}^n$, each configuration in the image set $y \in \Phi(X)$ can be computed from a singleton set $\{x\}$ for some $x \in \mathbb{B}^n$. In the following, we call such updates *set updates*.

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

▶ **Definition 8.** *Given a set update function $\Phi$ for BNs of dimension $n$, the generated transition relation is given by* $\delta : (2^{\mathbb{B}^n} \to 2^{\mathbb{B}^n}) \to 2^{\mathbb{B}^n \times \mathbb{B}^n}$ *with* $\delta(\Phi) = \{(x, y) \mid x \in \mathbb{B}^n, y \in \Phi(\{x\})\}$.

[2] The *fully-asynchronous* mode is usually referred to as *asynchronous* in the system biology modeling community.

In contrast with deterministic updates, non-deterministic updating modes can be characterized directly by set updates. Indeed, non-deterministic updating modes allow "superposing" alternative updates to generate different transitions from a single configuration $x$, although each of them is computed with a deterministic update. For instance, with one update $\phi$ where $\phi(x) = y$ and another update $\phi'$ where $\phi'(x) = y' \neq y$. Now, let us imagine an updating mode superposing two set updates, $\Phi$ and $\Phi'$ where, for some configurations $x \in \mathbb{B}^n$, $\Phi(\{x\}) \setminus \Phi'(\{x\}) \neq \emptyset$. One can then build a single set update $\Phi^*$ such that $\Phi^*(X) = \Phi(X) \cup \Phi'(X)$. It results that $\delta(\Phi^*) = \delta(\Phi) \cup \delta(\Phi')$, thus the updating mode can be assimilated to $\Phi^*$.

Finally, notice that limit sets of the generated dynamics $\delta(\Phi)$ can be characterized as the $\subseteq$-smallest sets of configurations $X \subseteq \mathbb{B}^n$ such that $\Phi(X) = X$.

## 5 Updating modes selecting elementary transitions

With deterministic updates as building blocks, we have seen that one can define non-deterministic updating modes by superposing different update functions. The resulting transition relation is then the union of the transition relation generated by each individual update (each of them giving a deterministic dynamics). Set updates offer an alternative way to formalize the resulting dynamics, by directly defining the set of out-going transitions from a given configuration. As we will illustrate with the memory updating mode below, this enables a fine-grained selection of the elementary transitions which may then depend on the configuration.

### 5.1 Asynchronous and fully-asynchronous updating modes

As a first illustration of set updates and how they can characterize updating modes, consider the following set update for BNs of dimension $n$:

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

This set update generates exactly all the elementary transitions: $\delta(\Phi_{\mathsf{e}}) = \rightarrow_{\mathsf{e}}$. Thus, $\Phi_{\mathsf{e}}$ characterizes the asynchronous updating mode. Similarly, let us now consider the following set update:

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

Remark that $\delta(\Phi_{\mathsf{fa}}) = \rightarrow_{\mathsf{fa}}$, i.e., $\Phi_{\mathsf{fa}}$ characterizes the fully-asynchronous updating mode.

### 5.2 Memory updating mode

Until now, all the updating modes that have been discussed depend on deterministic updates that are context free, which leads to deal with memoryless dynamical systems. In [14, 15] have been introduced another model of BNs, called Memory Boolean networks (MBNs). The first objective of MBNs is to capture the biologically relevant gene-protein BN model introduced in [18], that builds on the following principles:

- automata are split in two types: a half models genes, the other half models their associated one-to-one proteins;
- each protein has its own decay time: the number of time steps during which it remains present in the cell after having been produced by the punctual expression of its associated gene.

In their original definition given below, MBNs of dimension $n$ are BNs of dimension $n$ parameterized with a vector $\mathsf{M} \in \mathbb{N}^n_{>0}$, setting the maximal delay (called memory) for the degradation of each automaton. Then, an automaton is considered active (Boolean 1) whenever its delay to degradation is not 0. Formally, MBN are defined as follows:

▶ **Definition 9.** *A* Memory Boolean network *of dimension $n$ is the couple of a BN $f$ of dimension $n$ and of a memory vector $\mathsf{M} = (\mathsf{M}_1, \dots, \mathsf{M}_n) \in \mathbb{N}^n_{>0}$. The set of its configurations is defined as $X_{(f,\mathsf{M})} = \{(x,d) \in \mathbb{B}^n \times \mathbb{N}^n \mid \forall i \in [\![n]\!],\ d_i \in \{0,\dots,\mathsf{M}_i\},\ x_i = 0 \iff d_i = 0$ and $x_i = 1 \iff d_i \in \{1,\dots,\mathsf{M}_i\}\}$. The dynamical system $((f,\mathsf{M}),\mathsf{p})$ is defined by the transition graph $\mathcal{D}_{((f,\mathsf{M}),\mathsf{p})}$, with $\mathsf{p}$ the parallel updating mode, made of transitions based on updating function $\phi^\star : X_{(f,\mathsf{M})} \to X_{(f,\mathsf{M})}$ depending on the memories such that:*

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

*where* $\forall i \in [\![n]\!],\ \phi^\star_{[\![n]\!]}(x,d)_i = (y_i, d'_i)$, *with:*

$$d'_i = \begin{cases} 0 & \text{if } f_i(x) = 0 \text{ and } d_i = 0, \\ d_i - 1 & \text{if } f_i(x) = 0 \text{ and } d_i \geq 1, \\ \mathsf{M}_i & \text{if } f_i(x) = 1, \end{cases} \quad \text{and} \quad y_i = \begin{cases} 1 & \text{if } d'_i \geq 1, \\ f_i(x) & \text{if } d'_i = 0. \end{cases}$$

From this initial definition, it is easy to see that the dynamics of a MBN is deterministic and operates on discrete configurations that are not Boolean anymore. But we will see that MBNs enable to develop a new updating mode, called the memory updating mode, that operates directly on Boolean configurations.

First, let us define $\alpha(x)$ the set of memory configurations corresponding to any binary configuration $x \in \mathbb{B}^n$, and conversely, $\beta(d)$ the binary configuration corresponding to a memory configuration $d \in \mathbb{N}^n$. Notice that $\forall x \in \mathbb{B}^n$, $\forall d \in \alpha(x)$, $\beta(d) = x$.

$$\alpha(x) = \{d \in \mathbb{N}^n \mid x_i = 0 \Leftrightarrow d_i = 0, x_i = 1 \Leftrightarrow d_i \in [\![\mathsf{M}_i]\!]\},$$
$$\forall i \in [\![n]\!] \quad \beta(d)_i = \min\{d_i, 1\}.$$

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

$$\phi^*_{\mathsf{M}}(d)_i = \begin{cases} 0 & \text{if } f_i(\beta(d)) = 0 \text{ and } d_i = 0, \\ d_i - 1 & \text{if } f_i(\beta(d)) = 0 \text{ and } d_i \geq 0, \\ \mathsf{M}_i & \text{if } f_i(\beta(d)) = 1. \end{cases}$$

Let us now extend the above definitions to sets:

- $\forall X \subseteq \mathbb{B}^n$, $A(X) = \bigcup_{x \in X} \alpha(x)$;
- $\forall D \subseteq \mathbb{N}^n$, $B(D) = \{\beta(d) \mid d \in D\}$;
- $\forall D \subseteq \mathbb{N}^n$, $\Phi^*_{\mathsf{M}}(D) = \{\phi^*_{\mathsf{M}}(d) \mid d \in D\}$.

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

$$\Phi_{\mathsf{M}}(X) = B \circ \Phi^*_{\mathsf{M}} \circ A(X).$$

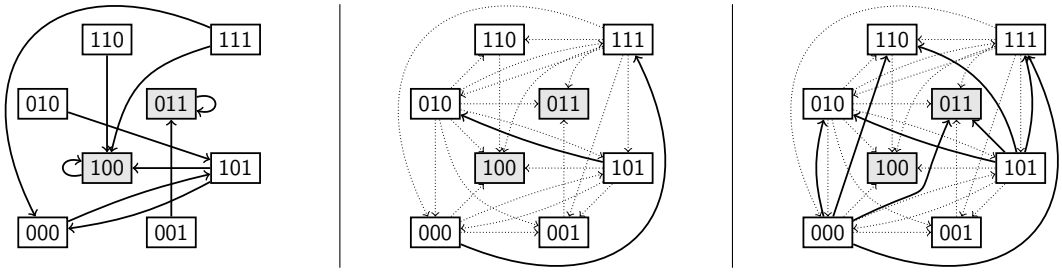



**Figure 3** Memory dynamics with $\overline{\mathsf{M}} = \{1\}$ (left), interval dynamics (center), and MP dynamics (right) of the BN of Example 2. In these two latter, the elementary transitions are dotted and loops are omitted.

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

$$\Phi_{\overline{\mathsf{M}}}(X) = \{\phi_W(x) \mid x \in X, W \subseteq [\![n]\!], W \supseteq \{i \in [\![n]\!] \mid i \notin \overline{\mathsf{M}} \vee f_i(x) = 1\}\}.$$

Remark that this definition no longer relies on memory configurations in $\mathbb{N}^n$. Overall, the memory updating mode of BNs can be understood as a particular set of elementary transitions: those where automata not in $\overline{\mathsf{M}}$ or automata that can change from state 0 to 1 are always updated, together with any subset of the others (automata in $\overline{\mathsf{M}}$ that can change from state 1 to 0): automata in $\overline{\mathsf{M}}$ that are decreasing are updated asynchronously, while the others are updated in parallel.

Figure 3(left) gives the dynamics generated by the interval updating mode on the BN of Example 2 with $\overline{\mathsf{M}} = \{1\}$.

## 6 Updating modes going beyond (non-)elementary transitions

BNs are widely used to model dynamics of biological systems, notably implying gene regulation. Gene regulation is a dynamical biological process that involves numerous mechanisms and entities among which some of them, like RNAs and proteins, have specific influences that depend on their concentration. In other terms, the regulation process in its whole admits significant quantitative parts. The question arises then of how faithful are Boolean dynamics with respect to the quantitative dynamics. It has been recently underlined in [24] that the elementary and non-elementary transitions of BNs are not complete enough to capture particular quantitative trajectories. With a fixed logic, and starting from similar configurations, the quantitative system shows that an automaton can eventually get activated, whereas the asynchronous dynamics of the BN shows it is impossible.

In this section, we address the set update reformulation of two recently introduced dynamics of BNs which generate transitions that are neither elementary nor non-elementary: they result in set updates $\Phi$ where, for some BNs of dimension $n$ and for some configurations $x \in \mathbb{B}^n$, there is $k \in \mathbb{N}$ such that there exists $y \in \Phi^k(\{x\})$ whereas $x \not\to_{\mathsf{e}}^* y$.

## 6.1 Interval updating mode

From the concurrency theory, it is known that the execution of 1-bounded contextual Petri nets corresponding to the asynchronous updating mode (known as *steps* semantics) may miss some transitions that can be triggered when considering certain delay of state change [6, 5]. In [7] is proposed a translation of the *interval* semantics of Petri nets to BNs, showing it can predict transitions that are neither elementary nor non-elementary transitions: configurations that are not reachable with the asynchronous mode from a fixed initial configuration $x$ become reachable with *Interval Boolean networks*.

The main principle of the interval dynamics is to decompose the change of state of an automaton, allowing interleaving the update of other automata. For instance, let us assume that automaton $i$ can change from state 0 to 1. In the interval dynamics, we register that $i$ will eventually change to state 1, and then allow the update of other automata, still considering that $i$ is in state 0. In [7], this is applied to any BN of dimension $n$ by an encoding with the fully-asynchronous dynamics of a BN of dimension $2n$: each automaton is split in a *read* and *write* automaton, where the write automaton register the next state of the original automaton, and the read automaton keeps its current state and will eventually copy the value of the write automaton. The dynamics of the original BN is then obtained by projecting the configurations on the read automata.

We provide below an equivalent formulation as a composition of set updates. Essentially, whenever an automaton $i$ can change its state, we hold it and compute the possible state changes of the automata different than $i$. Thus, during this evaluation, we have a growing number of held automata, that we denote by $L$, waiting for their state to be updated. The set update $\Phi_{\mathrm{Int},L}$ extends a given set of configurations with the possible state change of automata not in $L$. The function $\Phi_{i/L}(x)$ first computes all possible state changes by iterating $\Phi_{\mathrm{Int},L\cup\{i\}}$ until a fixed point, and then apply the state change for the automaton $i$ on all the resulting configurations.

▶ **Definition 10.** *The* Interval set update $\Phi_{\mathrm{Int}}$ *of a BN of dimension $n$ is given by* $\Phi_{\mathrm{Int}} = \Phi_{\mathrm{Int},\emptyset}$ *where*

$$\begin{aligned}\Phi_{\mathrm{Int},L}(X) &= X \cup \{y \in \Phi_{i/L}(x) \mid x \in X, i \in [\![n]\!], i \notin L, f_i(x) \neq x_i\},\\ \Phi_{i/L}(x) &= \{y^{\bar{i}} \mid y \in \Phi^{\omega}_{\mathrm{Int},L\cup\{i\}}(\{x\})\}.\end{aligned}$$

The interval updating mode preserves the fixed points of $f$: for any configuration $x \in \mathbb{B}^n$, $\Phi_{\mathrm{Int}}(\{x\}) = \{x\}$ if and only if $f(x) = x$. Moreover, one can prove that it includes all the elementary transitions: for any configuration $x \in \mathbb{B}^n$, $\Phi_{\mathsf{e}}(\{x\}) \subseteq \Phi_{\mathrm{Int}}(\{x\})$.

▶ **Example 11.** Figure 3 shows the transitions generated by the interval updating mode on the BN of Example 2. Notice that there is a path from 000 to 111, which does not exist in the asynchronous dynamics (Figure 2). Indeed, let us partially compute $\Phi_{\mathrm{Int},\emptyset}(\{000\}) = \{000\} \cup \Phi_{1/\emptyset}(000) \cup \Phi_{3/\emptyset}(000)$.

Let us focus on the interval update of automaton 1 with $\Phi_{1/\emptyset}(000)$, which requires computing all the iterations of $\Phi_{\mathrm{Int},\{1\}}(\{000\})$. The first iteration gives $\Phi_{\mathrm{Int},\{1\}}(\{000\}) = \{000\} \cup \Phi_{3/\{1\}}(000) = \{000, 001\}$, then $\Phi^2_{\mathrm{Int},\{1\}}(\{000\}) = \{000, 001\} \cup \Phi_{2/\{1\}}(001) = \{000, 001, 011\} = \Phi^{\omega}_{\mathrm{Int},\{1\}}(\{000\})$.

Finally, we get $\Phi_{1/\emptyset}(000) = \{100, 101, 111\}$. Thus, $111 \in \Phi_{\mathrm{Int}}(\{000\})$.

## 6.2 Most Permissive updating mode

Most Permissive Boolean networks (MPBNs) have been designed to capture all automata updates that could occur in any quantitative refinement of the BN. We will come back more formally to this notion later in this section.

The main feature of MPBNs is to abstract all the possible interaction thresholds between automata. Consider the case whenever the state of an automaton $i$ is used to compute the state of two distinct automata $j$ and $k$, and assume that $i$ is increasing from 0: during its increase, there are times when $i$ may be high enough for trigger a state change of $j$ but not (yet) high enough for $k$. This can be illustrated on a concrete biological example, the so-called *incoherent feed-forward loop of type 3* [20]: a BN $f$ of dimension 3 with

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

Starting from the configuration 000, the asynchronous updating mode predicts only the following non-reflexive transitions: $000 \to_{\mathsf{a}} 100 \to_{\mathsf{a}} 110$. Notice that in this case, the interval updating mode results in the same transitions. However, it has been observed experimentally [27] and in quantitative models [19, 26] that depending on reaction kinetics, one can actually activate transiently the automata 3. Essentially, the idea is that during the increase of the state of automaton 1, there is period of time where 1 is high enough so 2 can consider it active ($x_1$ true) but 3 still considers it inactive ($x_1$ false). Then, the state of automaton 2 can increase, and so do the state of automaton 3. This activation of 3 cannot be predicted with BN updating modes defined so far, whereas the logic encoded by $f$ is correct.

Without introducing any parameter, MPBNs capture these additional dynamics by accounting for all possible thresholds ordering, for all updates that can happen between a switch of a Boolean state. In some sense, the MP updating mode abstracts both the quantitative domain of automata and the duration of state changes. Their original definition [24] is based on the introduction pseudo *dynamic* states, namely increasing and decreasing. An automaton can change from 0 to increasing whenever it can interpret the state of the other automata so that its local function is satisfied. Once in increasing state, it can change to the state 1 without any condition, or to the decreasing state whenever it can interpret the state of other automata so that its local function is not satisfied. Whenever an automaton is in a dynamic state, the automata can freely interpret its state as either 0 or 1. Remark that the possible interpretations of the MP configurations always result in a hypercube (a set of automata fixed to a Boolean value, and the others free).

Here, we show that the MP dynamics can be expressed in a more standard way by the means of composition of set updates. A first stage consists in *widening* all the elementary set updates to compute all the possible interpretations of automata changing of state. The widening is defined using the function $\nabla : 2^{\mathbb{B}^n} \to 2^{\mathbb{B}^n}$ which computes the vertices of the smallest hypercube containing the given set of configurations. For instance, $\nabla(\{01, 10\}) = \{00, 01, 10, 11\}$. Given a set of automata $W$, the widening set update $\Phi_{W,\nabla} : 2^{\mathbb{B}^n} \to 2^{\mathbb{B}^n}$ applies this operator on the results of the elementary set update, or equivalently with the fully-asynchronous set update, on the automata of $W$ (Subsection 5.1). This widening is re-iterated until a fixed point is reached. Then, a *narrowing* $\Lambda_W : 2^{\mathbb{B}^n} \to 2^{\mathbb{B}^n}$ filters the computed configurations $X$ to retain only those where the states of automata in $W$ can be computed with $f$ from $X$.

▶ **Definition 12.** *The* Most Permissive set update $\Phi_{\mathsf{MP}}$ *of a BN of dimension $n$ is given by*

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

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

$$\nabla(X) = \{x \in \mathbb{B}^n \mid \forall i \in [\![n]\!], \exists y \in X : x_i = y_i\}, \tag{1}$$

$$\Phi_{W,\nabla}(X) = \nabla(X \cup \{\phi_i(x) \mid x \in X, i \in W\}), \tag{2}$$

$$\Lambda_W(X) = \{x \in X \mid \forall i \in W, \exists y \in X : x_i = f_i(y)\}. \tag{3}$$

▶ **Example 13.** Figure 3 shows the dynamics generated by the MP updating mode on the BN of Example 2. With the BN $f$ of the incoherent feed-forward loop introduced at the beginning of Subsection 6.2 page 10:11, we obtain:

$$\begin{aligned}
\Phi_{\{1,2,3\},\nabla}(\{000\}) &= \nabla(\{000,100\}) = \{000,100\},\\
\Phi^2_{\{1,2,3\},\nabla}(\{000\}) &= \nabla(\{000,100\}\cup\{110\}) = \{000,100,010,110\},\\
\Phi^3_{\{1,2,3\},\nabla}(\{000\}) &= \nabla(\{000,1000,010,110\}\cup\{011\}) = \mathbb{B}^n,\\
\Lambda_{\{1,2,3\}}(\mathbb{B}^n) &= \{100,101,110,111\}.
\end{aligned}$$

Thus, $111 \in \Phi_{\mathsf{MP}}(\{000\})$, whereas $000 \not\to_{\mathsf{e}}^* 111$ and $111 \notin \Phi_{\mathrm{Int}}(\{000\})$.

Let us now list some basic properties of the MP updating mode:

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

In [24], it has been demonstrated that MP dynamics of a BN $f$ forms a correct abstraction of the dynamics of any quantitative model being a *refinement* of $f$. A quantitative model $F$ can be defined as a function mapping discrete or continuous configurations to the derivative of the state of automata. Then, $F$ is a refinement of $f$ if and only if the derivative of automaton $i$ is strictly positive (resp. negative) in a given quantitative configuration $z$ only if there is a binarization $\tilde{z}$ of $z$ so that $f_i(\tilde{z}) = 1$ (resp. 0). It has also been proven to be minimal for the abstraction of asynchronous discrete models. Moreover, the complexity for deciding the existence of a path between two configurations as well as deciding whether a configuration belongs to a limit set is respectively in $\mathrm{P}^{\mathrm{NP}}$ and in $\mathrm{coNP}^{\mathrm{coNP}}$ in general and in P and in coNP for locally monotonic BNs (each local function is monotonic with respect to a specific component-wise ordering of configurations), in contrast with the other updating modes where these problems are PSPACE-complete.

## 7 Discussion

By extending to non-deterministic updates modeled as set updates, we can reformulate in a unified manner a range of BN dynamics introduced in the literature with ad-hoc definitions, and for which the usual deterministic updates seem not expressive enough. These reformulations bring a better understanding and comparison of dynamics as more classical BN updating modes. Moreover, they allow envisioning new families of updating modes as variations of the one presented here. For instance, the given MP set update allows to readily define restrictions of it: similarly to the block-sequential updating mode, one could parameterize the MP set update to only consider particular sequences of sets of automata to update. One could also consider different narrowing operators and different manners to compose them with the widening, with the goal of reducing the set of generated transitions.

On the one hand, these set updates foster the definitions of totally new kinds of updating modes. On the other hand, they raise the question of a potential upper limit on which transitions could be considered as valid, or at least reasonable.

### On *reasonable* set updates

Of course, from a purely theoretical standpoint, any set update which is mathematically correct is reasonable but, if we consider set updates in a context of modeling, some constraints need to be taken in account. This second standpoint is the one on which is based the following discussion. Indeed, as evoked in the introduction of this paper, BNs are a classical mathematical model in systems biology. They are notably widely used to model genetic regulation networks, in which their use rests for instance on the fact that their limit sets model real observable "structures" such as differentiated cellular types (fixed points), or specific biological paces (limit cycles). In this sense, a basic criterion would be that an updating mode for a BN $f$ is admissible only if the fixed points of $f$ are fixed points of the generated dynamics as well. This criterion would allow capturing the fundamental property of fixed point stability of dynamical system theory. For instance, let us consider the set update $\Phi_\top(X) = \mathbb{B}^n$: clearly, the set of fixed points of the generated dynamical system is always empty, and thus do not include those of $f$ whenever $f$ has at least one fixed point. Therefore, such a set update does not appear satisfying.

Now, let us discuss about set updates which would give sets larger than MP for some singleton configuration set $\{x\}$. First, what about defining a widening operator larger than $\nabla$? For any set of automata $W$ and for any configuration $x$, remark that $\Phi^\omega_{W,\nabla}(\{x\}) = Y$ is the smallest hypercube containing $x$ verifying for each automata $i \in W$ that for any configuration $y \in Y$, if $f_i(y) \neq x_i$, then there exists a configuration $z \in Y$ with $z_i \neq x_i$. Thus, an automaton in $W$ is either fixed to its state in $x$, or it has been computed with its local function from at least one configuration from a smaller hypercube. Therefore, a widening operator $\nabla'$ verifying for some $X \subseteq \mathbb{B}^n$, $\nabla'(X) \supsetneq \nabla(X)$ implies that the state of at least one configuration is not computed using $f$ on $X$. Now, what about a less stringent narrowing operator. Let us consider a configuration $y \in \Phi_{W,\nabla}(\{x\}) = Y$ for some set of automata $W$, but $y \notin \Lambda_W(Y)$. This implies that there exists an automaton $i \in W$ such that $\forall z \in Y$, $y_i \neq f_i(z)$, i.e., $y_i$ cannot be computed by $f_i$ from $X$. Overall, a set update giving configuration sets strictly larger than the MP update implies that for some configurations, the state of at least one automaton is not computed using its local function.

### Simulations by deterministic updates

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

Actually, a fundamental matter here lies in the concept of simulation at stake here: we are interested in intrinsic simulations which go far beyond the classical concepts of encoding or simulation. Indeed, intrinsic simulations aim at conserving dynamical structures

in addition to operated computations. So, one of the first question to answer would consist in defining formally different kinds of intrinsic simulations. Nevertheless, firstly, consider the following intrinsic simulation: a dynamical system $(f, \mu)$ simulates another $(g, \mu')$ if $\mathcal{D}_{(g,\mu')}$ is a subgraph of $\mathcal{D}_{(f,\mu)}$. With this rather simple definition, it is direct to state that, with $\mathsf{a}$ and $\overline{\mathsf{M}}$ the asynchronous and memory updating modes respectively, for any BN $f$, $(f, \mathsf{a})$ simulates $(f, \overline{\mathsf{M}})$. Some natural questions related to BNs updated with memory are the following:

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

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

- Let $\mathsf{per}$ be a deterministic periodic updating mode. How can $(f, \overline{\mathsf{M}})$ be simulated by $(g, \mathsf{per})$? The answer is known for $\mathsf{per} = \mathsf{p}$ [14], but it seems pertinent to find a generalization to deterministic periodic updating modes, and even more general deterministic updating modes.
- Intuitively, any $(f, \mathsf{MP})$ might be simulated by $(g, \mathsf{a})$, where $f$ and $g$ are BNs and the dimension of $g$ is greater than that of $f$. But how many automata need to be added to $g$ depending on the dimension of $f$?

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

## References


1 Julio Aracena. Maximum number of fixed points in regulatory Boolean networks. *Bulletin of Mathematical Biology*, 70:1398–1409, 2008. doi:10.1007/s11538-008-9304-7.

2 Julio Aracena, Eric Fanchon, Marco Montalva, and Mathilde Noual. Combinatorics on update digraphs in Boolean networks. *Discrete Applied Mathematics*, 159:401–409, 2011. doi:10.1016/j.dam.2010.10.010.

3 Julio Aracena, Eric Goles, Andres Moreira, and Lilian Salinas. On the robustness of update schedules in Boolean networks. *Biosystems*, 97:1–8, 2009. doi:10.1016/j.biosystems.2009.03.006.

4 Florian Bridoux, Caroline Gaze-Maillot, Kévin Perrot, and Sylvain Sené. Complexity of limit-cycle problems in Boolean networks. In *Proceedings of the International Conference on Current Trends in Theory and Practice of Informatics (SOFSEM'21)*, volume 12607 of *Lecture Notes in Computer Science*, pages 135–146. Springer, 2021. doi:10.1007/978-3-030-67731-2_10.

5 Thomas Chatain, Stefan Haar, Juraj Kolčák, Loïc Paulevé, and Aalok Thakkar. Concurrency in Boolean networks. *Natural Computing*, 19(1):91–109, 2020. doi:10.1007/s11047-019-09748-4.

6 Thomas Chatain, Stefan Haar, Maciej Koutny, and Stefan Schwoon. Non-atomic transition firing in contextual nets. In *Applications and Theory of Petri Nets*, volume 9115 of *Lecture Notes in Computer Science*, pages 117–136. Springer, 2015. doi:10.1007/978-3-319-19488-2_6.

7 Thomas Chatain, Stefan Haar, and Loïc Paulevé. Boolean Networks: Beyond Generalized Asynchronicity. In *Cellular Automata and Discrete Complex Systems (AUTOMATA 2018)*, volume 10875 of *Lecture Notes in Computer Science*, pages 29–42, Ghent, Belgium, 2018. Springer. doi:10.1007/978-3-319-92675-9_3.

**8** Michel Cosnard and Jacques Demongeot. On the definitions of attractors. In *Proceedings of the International Symposium on Iteration Theory and its Functional Equations*, volume 1163 of *Lecture Notes in Mathematics*, pages 23–31. Springer, 1985. doi:10.1007/BFb0076414.

**9** Jacques Demongeot, Eric Goles, Michel Morvan, Mathilde Noual, and Sylvain Sené. Attraction basins as gauges of the robustness against boundary conditions in biological complex systems. *PLoS One*, 5:e11793, 2010. doi:10.1371/journal.pone.0011793.

**10** Jacques Demongeot, Mathilde Noual, and Sylvain Sené. Combinatorics of Boolean automata circuits dynamics. *Discrete Applied Mathematics*, 160:398–415, 2012. doi:10.1016/j.dam.2011.11.005.

**11** Jacques Demongeot and Sylvain Sené. About block-parallel Boolean networks: a position paper. *Natural Computing*, 19:5–13, 2020. doi:10.1007/s11047-019-09779-x.

**12** A. Elena. *Robustesse des réseaux d'automates booléens à seuil aux modes d'itération. Application à la modélisation des réseaux de régulation génétique*. PhD thesis, Université Grenoble 1 – Joseph Fourier, 2009. URL: https://tel.archives-ouvertes.fr/tel-00447564/document.

**13** Françoise Fogelman, Eric Goles, and Gérard Weisbuch. Transient length in sequential iteration of threshold functions. *Discrete Applied Mathematics*, 6:95–98, 1983. doi:10.1016/0166-218X(83)90105-1.

**14** Eric Goles, Fabiola Lobos, Gonzalo A. Ruz, and Sylvain Sené. Attractor landscapes in Boolean networks with firing memory: a theoretical study applied to genetic networks. *Natural Computing*, 19:295–319, 2020. doi:10.1007/s11047-020-09789-0.

**15** Eric Goles, Pedro Montealegre, and Martín Ríos-Wilson. On the effects of firing memory in the dynamics of conjunctive networks. In *Proceedings of the International Workshop on Cellular Automata and Discrete Complex Systems (AUTOMATA'19)*, volume 11525 of *Lecture Notes in Computer Science*, pages 1–19. Springer, 2019. doi:10.1007/978-3-030-20981-0_1.

**16** Eric Goles and Mathilde Noual. Block-sequential update schedules and Boolean automata circuits. In *Proceedings of the International Workshop on Cellular Automata and Discrete Complex Systems (AUTOMATA'10)*, pages 41–50. Discrete Mathematics and Theoretical Computer Science, 2010. URL: https://hal.inria.fr/hal-01185498/document.

**17** Eric Goles and Lilian Salinas. Comparison between parallel and serial dynamics of Boolean networks. *Theoretical Computer Science*, 396:247–253, 2008. doi:10.1016/j.tcs.2007.09.008.

**18** Alex Graudenzi and Roberto Serra. A new model of genetic networks: the gene protein Boolean network. In *Proceedings of the Workshop italiano su vita artificiale e calcolo evolutivo (WIVACE'09)*, pages 283–291. World Scientific, 2009. doi:10.1142/9789814287456_0025.

**19** Shuji Ishihara, Koichi Fujimoto, and Tatsuo Shibata. Cross talking of network motifs in gene regulation that generates temporal pulses and spatial stripes. *Genes to Cells*, 10(11):1025–1038, 2005. doi:10.1111/j.1365-2443.2005.00897.x.

**20** S. Mangan and U. Alon. Structure and function of the feed-forward loop network motif. *Proceedings of the National Academy of Sciences*, 100(21):11980–11985, 2003. doi:10.1073/pnas.2133841100.

**21** John Milnor. On the concept of attractor. *Communications in Mathematical Physics*, 99:177–185, 1985. doi:10.1007/978-0-387-21830-4_15.

**22** Mathilde Noual. *Updating automata networks*. PhD thesis, École normale supérieure de Lyon, 2012. URL: https://tel.archives-ouvertes.fr/tel-00726560/document.

**23** Mathilde Noual and Sylvain Sené. Synchronism versus asynchronism in monotonic Boolean automata networks. *Natural Computing*, 17:393–402, 2018. doi:10.1007/s11047-016-9608-8.

**24** Loïc Paulevé, Juraj Kolčák, Thomas Chatain, and Stefan Haar. Reconciling qualitative, abstract, and scalable modeling of biological networks. *Nature Communications*, 11(1), 2020. doi:10.1038/s41467-020-18112-5.

**25** François Robert. *Discrete iterations: a metric study*, volume 6 of *Springer Series in Computational Mathematics*. Springer, 1986. doi:10.1007/978-3-642-61607-5.

**26** Guillermo Rodrigo and Santiago F. Elena. Structural discrimination of robustness in transcriptional feedforward loops for pattern formation. *PLoS One*, 6(2):e16904, 2011. doi:10.1371/journal.pone.0016904.

**27** Yolanda Schaerli, Andreea Munteanu, Magüi Gili, James Cotterell, James Sharpe, and Mark Isalan. A unified design space of synthetic stripe-forming networks. *Nature Communications*, 5(1), 2014. doi:10.1038/ncomms5905.

# Von Neumann Regularity, Split Epicness and Elementary Cellular Automata

**Ville Salo**
University of Turku, Finland

## Abstract

We show that a cellular automaton on a mixing subshift of finite type is a von Neumann regular element in the semigroup of cellular automata if and only if it is split epic onto its image in the category of sofic shifts and block maps. It follows from [S.-Törmä, 2015] that von Neumann regularity is decidable condition, and we decide it for all elementary CA.

**2012 ACM Subject Classification** Theory of computation → Automata over infinite objects

**Keywords and phrases** cellular automata, elementary cellular automata, von Neumann regularity, split epicness

**Digital Object Identifier** 10.4230/OASIcs.AUTOMATA.2021.11

**Related Version** *Previous Version*: `https://arxiv.org/abs/1804.03913`

**Funding** *Ville Salo*: Research funded by Academy of Finland grant 2608073211.

**Acknowledgements** We thank Jarkko Kari for observing that Proposition 12 works in all dimensions. We thank Johan Kopra for pointing out that Lemma 11 is easier to prove than to find in [6].

## 1 Introduction

The von Neumann regular elements – elements $a$ having a weak inverse $b$ such that $aba = a$ – of cellular automaton (CA) semigroups are studied in [1]. We show that in the context of cellular automata on mixing subshifts of finite type, von Neumann regularity coincides with the notion of split epicness onto the image, another generalized invertibility notion from category theory.

Question 1 of [1] asks which of the so-called elementary cellular automata (ECA) are von Neumann regular. They determine this for all ECA except ones equivalent to those with numbers 6, 7, 9, 23, 27, 28, 33, 41, 57, 58 and 77, see the next section for the definition of the numbering scheme.

What makes this question interesting is that von Neumann regularity of one-dimensional cellular automata is not obviously[1] decidable – clearly checking if $g$ is a weak inverse is semidecidable, but it is not immediately clear how to semidecide the nonexistence of a weak inverse. However, split epicness has been studied previously in [9], and in particular it was shown there that split epicness of a morphism between two sofic shifts is a decidable condition. This means Question 1 of [1] can in theory be decided algorithmically.

As the actual bound stated in [9] is beyond astronomical, it is an interesting question whether the method succeeds in actually deciding each case. With a combination of this method, computer and manual searches, and some ad hoc tricks, we prove that ECA 6, 7, 23, 33, 57 and 77 are von Neumann regular, while 9, 27, 28, 41 and 58 are not, answering the remaining cases of Question 1 of [1].

[1] Specifically, many things about "one-step behavior" of cellular automata (like surjectivity and injectivity) are decidable using automata theory, or the decidability of the MSO of the natural numbers under successor. No decision algorithm for split epicness using these methods is known.

27th IFIP WG 1.5 International Workshop on Cellular Automata and Discrete Complex Systems (AUTOMATA 2021).
Editors: Alonso Castillo-Ramirez, Pierre Guillon, and Kévin Perrot; Article No. 11; pp. 11:1–11:10

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

The von Neumann regular CA on this list have weak inverses of radius at most five. Non-regularity is proved in each case by looking at eventually periodic points of eventual period one. The non-regularity of all but ECA 9 and ECA 28 can be proved by simply observing that their images are proper sofic, though we also explain why they are not regular using the method of [9].

## 2 Preliminaries

The *full shift* is $\Sigma^{\mathbb{Z}}$ where $\Sigma$ is a finite alphabet, carrying the product topology, It is a dynamical system under the *shift* $\sigma(x)_i = x_{i+1}$. Its subsystems (closed shift-invariant subsets) are called *subshifts*. A *cellular automaton (CA)* is a shift-commuting continuous function $f : X \to X$ on a subshift $X$. The cellular automata on a subshift $X$ form a monoid $\mathrm{End}(X)$. A CA $f$ is *reversible* if $\exists g : f \circ g = g \circ f = \mathrm{id}$.

A cellular automaton has a local rule, that is, there exists a *radius* $r \in \mathbb{N}$ such that $f(x)_i$ is determined by $x|_{[i-r,i+r]}$ for all $x \in X$ (and does not depend on $i$). The *elementary cellular automata* (ECA) are the CA on the binary full shift $\{0,1\}^{\mathbb{Z}}$ which can be defined with radius 1. There is a numbering scheme for such CA: If $n \in [0, 255]$ has base 2 representation $b_7b_6...b_1b_0$, then ECA number $n$ is the one mapping

$$f(x)_i = 1 \iff b_{(x_{[i-1,i+1]})_2} = 1$$

where $(x_{[i-1,i+1]})_2$ is the number represented by $x_{[i-1,i+1]}$ in base 2. This numbering scheme is from [11].

We recall [1, Definition 3]: define maps $R, S : \{0,1\}^{\mathbb{Z}} \to \{0,1\}^{\mathbb{Z}}$ by the formulas $R(x)_i = x_{-i}$ and $S(x)_i = 1 - x_i$. Two cellular automata $f, g \in \mathrm{End}(\{0,1\}^Z)$ are *equivalent* if $f \in \langle S\rangle \circ g \circ \langle S\rangle \cup \langle S\rangle \circ R \circ g \circ R \circ \langle S\rangle$, where $\circ$ denotes function composition and $\langle S\rangle = \{\mathrm{id}, S\}$.

The usage of base 10 in this notation is standard, and many CA researchers remember ECA by these numbers. However, for clarity we switch to hexadecimal notation from radius 2 upward.

A subshift can be defined by forbidding a set of finite words from appearing as subwords of its points (which themselves are infinite words), and this is in fact a characterization of subshifts. A subshift is *of finite type* or *SFT* if it can be defined by a finite set of forbidden patterns, and *sofic* if it can be defined by a forbidden regular language in the sense of automata and formal languages.

The *language* $\mathcal{L}(X)$ of a subshift $X$ is the set of finite words that appear in its points. An SFT $X$ is *mixing* if $L = \mathcal{L}(X)$ satisfies $\exists m : \forall u, v \in L : \exists w \in L : |w| = m \wedge uwv \in L$.

If $u \in A^*$ is a finite word, we write ${}^{\infty}u^{\infty}$ for the $|u|$-periodic point (i.e. fixed point of $\sigma^{|u|}$) in $A^{\mathbb{Z}}$ whose subword at $\{0, 1, ..., |u| - 1\}$ is equal to $u$.

See standard references for more information on symbolic dynamics [6] or automata theory [4].

## 3 Split epicness and von Neumann regularity

In this section, we show split epicness and von Neumann regularity are equivalent concepts on mixing SFTs. On the full shift, this is simply a matter of defining these terms.

If $S$ is a semigroup, then $a \in S$ is *(von Neumann) regular* if $\exists b \in S : aba = a \wedge bab = b$. We say $b$ is a *generalized inverse* of $a$. If $aba = a$, then $b$ is a *weak generalized inverse* of $a$.

▶ **Lemma 1.** *If $a$ has a weak generalized inverse, then it has a generalized inverse and thus is regular.*

**Proof.** If $aba = a$, then letting $c = bab$, we have $aca = ababa = aba = a$ and $cac = bababab = babab = bab = c$. ◀

If $\mathcal{C}$ is a category, a morphism $f : X \to Y$ is *split epic* if there is a morphism $g : Y \to X$ such that $f \circ g = \mathrm{id}_Y$. Such a $g$ is called a *right inverse* or a *section*.

Note that in general category-theoretic concepts depend on the particular category at hand, but if $\mathcal{C}$ is a full subcategory of $\mathcal{D}$ (meaning a subcategory induced by a subclass of the objects, by taking all the morphisms between them), then split epicness for a morphism $f : X \to Y$ where $X, Y$ are objects of $\mathcal{C}$ means the same in both.

We are in particular interested in the category $K3$ (in the naming scheme of [9]) with sofic shifts as objects, and *block maps*, i.e. shift-commuting continuous functions $f : X \to Y$ as morphisms. More generally, morphisms between general subshifts have the same definition.

The following theorem is essentially only a matter of translating terminology, and works in many concrete categories.

▶ **Theorem 2.** *Let $X$ be a subshift, and $f : X \to X$ a cellular automaton. Then the following are equivalent:*

- *$f : X \to f(X)$ has a right inverse $g : f(X) \to X$ which can be extended to a morphism $h : X \to X$ such that $h|_{f(X)} = g$,*
- *$f$ is regular as an element of* $\mathrm{End}(X)$.

**Proof.** Suppose first that $f$ is regular, and $h \in \mathrm{End}(X)$ satisfies $fhf = f$ and $hfh = h$. Then the restriction $g = h|_{f(X)} : f(X) \to X$ is still shift-commuting and continuous, and $\forall x : fg(f(x)) = f(x)$ implies that for all $y \in f(X)$, $fg(y) = y$, i.e. $g$ is a right inverse for the codomain restriction $f : X \to f(X)$ and it extends to the map $h : X \to X$ by definition.

Suppose then that $fg = \mathrm{id}_{f(X)}$ for some $g : f(X) \to X$, as a right inverse of the codomain restriction $f : X \to f(X)$. Let $h : X \to X$ be such that $h|_{f(X)} = g$, which exists by assumption. Then $fh(f(x)) = fg(f(x)) = f(x)$. Thus $f$ is regular, and $hfh$ is a generalized inverse for it. ◀

Note that when $X$ is a full shift, extending morphisms is trivial: simply fill in the local rule arbitrarily. The Extension Lemma generalizes this idea to mixing SFTs:

▶ **Theorem 3.** *Let $X$ be a mixing SFT, and $f : X \to X$ a cellular automaton. Then the following are equivalent:*

- *$f : X \to f(X)$ is split epic in $K3$.*
- *$f$ is regular as an element of* $\mathrm{End}(X)$.

**Proof.** It is enough to show that any right inverse $g : f(X) \to X$ can be extended to $h : X \to X$ such that $h|_{f(X)} = g$. By the Extension Lemma [6], it is enough to show the "$X \searrow X$ condition" [6], which means that for every point $x \in X$ with minimal period $p$, there is a point $y \in X$ with minimal period dividing $p$. This holds trivially. ◀

Theorem 2 clearly implies that regularity respects equivalence (this is not difficult to obtain directly from the definition either).

▶ **Corollary 4.** *if $f, g \in \mathrm{End}(\{0,1\}^{\mathbb{Z}})$ are equivalent, then $f$ is regular if and only if $g$ is regular.*

## 4 Deciding split epicness

We recall the characterization of split epicness [9, Theorem 1]. This is Theorem 7 below.

▶ **Definition 5.** *Let $X, Y$ be subshifts and let $f : X \to Y$ be a morphism. Define*

$$\mathcal{P}_p(Y) = \{u \in \mathcal{L}(Y) \mid {}^{\infty}u^{\infty} \in Y, |u| \leq p\}.$$

*We say $f$ satisfies the* strong $p$-periodic point condition *if there exists a length-preserving function $G : \mathcal{P}_p(Y) \to \mathcal{L}(X)$ such that for all $u, v \in \mathcal{P}_p(Y)$ and $w \in \mathcal{L}(Y)$ with ${}^{\infty}u.wv^{\infty} \in Y$, there exists an $f$-preimage for ${}^{\infty}u.wv^{\infty}$ of the form ${}^{\infty}G(u)w'.w''w'''G(v)^{\infty} \in X$ where $|u|$ divides $|w'|$, $|v|$ divides $|w'''|$ and $|w| = |w''|$. The* strong periodic point condition *is that the strong $p$-periodic point condition holds for all $p \in \mathbb{N}$.*

Note that $G$ is simply a notation for a choice of periodic preimage for each periodic point, and the condition simply states that periodic tails of eventually periodic points eventually map according to $G$.

The strong periodic point condition is an obvious necessary condition for having a right inverse, as the right inverse must consistently pick preimages for periodic points, and they must satisfy these properties. Let us show the XOR CA with neighborhood $\{0, 1\}$ is not regular using this method – this is clear from the fact it is surjective, and from the fact there are 1-periodic points with no inverse of period 1, but it also neatly illustrates the strong periodic point method.

▶ **Example 6.** The CA $f : \{0, 1\}^{\mathbb{Z}} \to \{0, 1\}^{\mathbb{Z}}$ defined by

$$f(x)_i = 1 \iff x_i + x_{i+1} \equiv 1 \bmod 2$$

is not regular. To see this, consider the strong $p$-periodic point condition for $p = 1$. Since $f(0^{\mathbb{Z}}) = f(1^{\mathbb{Z}}) = 0^{\mathbb{Z}}$, the point $0^{\mathbb{Z}}$ has two preimages, and we must have either $G(0) = 0$ or $G(0) = 1$. It is enough to show that neither choice of $a = G(0)$ is consistent, i.e. there is a point $y$ which is in the image of $f$ such that $y$ has no preimage that is left and right asymptotic to $a^{\mathbb{Z}}$. This is shown by considering the point

$$y = ...0000001000000...$$

(which is in the image of $f$ since $f$ is surjective). It has two preimages, and the one left-asymptotic to $a^{\mathbb{Z}}$ is right-asymptotic to $(1 - a)^{\mathbb{Z}}$. ○

In [9, Theorem 1], it is shown that the strong periodic point condition actually characterizes split epicness, in the case when $X$ is an SFT and $Y$ is a sofic shift.

▶ **Theorem 7.** *Given two objects $X \subset S^{\mathbb{Z}}$ and $Y \subset R^{\mathbb{Z}}$ and a morphism $f : X \to Y$ in K3, it is decidable whether $f$ is split epic. If $X$ is an SFT, split epicness is equivalent to the strong periodic point condition.*

We note that Definition 5 is equivalent to a variant of it where $G$ is only defined on Lyndon words [7], i.e. lexicographically minimal representative words of periodic orbits: if $G$ is defined on those, it can be extended to all of $\mathcal{P}_p$ in an obvious way, and the condition being satisfied by minimal representatives implies it for all eventually periodic points.

▶ Remark 8. It is observed in [1, Theorem 1] that if $f : X \to Y$ is split epic, then every periodic point in $Y$ must have a preimage of the same period in $X$ – this is a special case of the above, and could thus be called the *weak periodic point condition*. In [9, Example 5],

an example is given of morphism between mixing SFTs which satisfies the weak periodic point condition but not the strong one. We have not attempted to construct an example of a CA on a full shift which has this property onto its image, and we did not check whether any non-regular ECA satisfies it. In [1, Theorem 4], for full shifts on finite groups, the weak periodic point condition is shown to be equivalent to split epicness (when CA are considered to be morphisms onto their image). In the context of CA on $\mathbb{Z}^2$, there is no useful strong periodic point condition in the sense that split epicness is undecidable, see Corollary 13.

In the proof of Theorem 7 in [9], decidability is obtained from giving a bound on the radius of a minimal inverse, and a very large one is given, as we were only interested in the theoretical decidability result. The method is, however, quite reasonable in practise:

- To semidecide non-(split epicness), look at periodic points one by one, and try out different possible choices for their preimages. Check by automata-theoretic methods (or "by inspection") which of these are consistent in the sense of Definition 5.
- To semidecide split epicness, invent a right inverse – note that here we can use the other semialgorithm (running in parallel) as a tool, as it tells us more and more information about how the right inverse must behave on periodic points, which tells us more and more values of the local rule.

One of these is guaranteed to finish eventually by [9].

Proposition 10 below is a slight generalization of [9, Proposition 1]. We give a proof here, as the proof in [9] unnecessarily applies a more difficult result of S. Taati (and thus needs the additional assumption of "mixing"). This Proposition allows us to obtain non-regularity of all of the non-regular ECA considered here apart from ECA 9 and 28, though we do also provide a strong periodic point condition argument for all the non-regular ECA.

▶ **Lemma 9.** *If $X$ is an SFT and $f : X \to X$ is idempotent, i.e. $f^2 = f$, then $f(X)$ is an SFT.*

**Proof.** Clearly $x \in f(X) \iff f(x) = x$, which is an SFT condition. ◀

▶ **Proposition 10.** *If $X$ is an SFT and $f : X \to X$ is regular, then $f(X)$ is of finite type.*

**Proof.** Let $g : X \to X$ be a weak inverse. Then $g \circ f : X \to X$ is idempotent, so $g(f(X))$ is an SFT. Note that the domain-codomain restriction $g|_{f(X),g(f(X))} : f(X) \to g(f(X))$ is a conjugacy between $f(X)$ and $g(f(X))$: its two-sided inverse is $f|_{g(f(X))} \to f(X)$ by a direct computation. Thus $f(X)$ is also an SFT. ◀

We also mention another condition, although it is not applicable in the proofs.

▶ **Lemma 11.** *Let $X$ be a subshift with dense periodic points and $f : X \to X$ a cellular automaton. If $f$ is injective, it is surjective.*

**Proof.** The set $X_p = \{x \in X \mid \sigma^p(x) = x\}$ satisfies $f(X_p) \subset X_p$. Since $f$ is injective and $X_p$ is finite, we must have $f(X_p) = X_p$. Thus $f(X)$ is a closed set containing the periodic points. If periodic points are dense, $f(X) = X$. ◀

We are interested mainly in mixing SFTs, where periodic points are easily seen to be dense. We remark in passing that in the case of mixing SFTs, the previous lemma can also be proved with an entropy argument: An injective CA cannot have a *diamond*[2] when seen as

[2] This means a pair of distinct words whose long prefixes and suffixes agree, and which the local rule maps the same way, see [6].

a block map, so [6, Theorem 8.1.16] shows that the entropy of the image $f(X)$ of an injective CA is equal to the entropy of $X$. By [6, Corollary 4.4.9], $X$ is *entropy minimal*, that is, has no proper subshifts of the same entropy, and it follows that $f(X) = X$.

▶ **Proposition 12.** *Let $X$ be a mixing SFT and $f : X \to X$ a surjective cellular automaton. Then $f$ is injective if and only if it is regular.*

**Proof.** Suppose $f$ is a surjective CA on a mixing SFT. If it is also injective, it is thus bijective, thus reversible, thus regular. Conversely, let $f$ be surjective and regular, and let $g : X \to X$ be a weak generalized inverse. Then $g$ is injective, so it is surjective by the previous lemma. Thus $f$ must be bijective as well. ◀

More generally, the previous proposition works on *surjunctive subshifts* in the sense of [2, Exercise 3.29], i.e., subshifts where injective cellular automata are surjective. In particular this is the case for full shifts on surjunctive groups [3, 10] such as abelian ones. Since injectivity is undecidable for surjective CA on $\mathbb{Z}^d$, $d \geq 2$ by [5], we obtain the following corollary.

▶ **Corollary 13.** *Given a surjective CA $f : \Sigma^{\mathbb{Z}^2} \to \Sigma^{\mathbb{Z}^2}$, it is undecidable whether $f$ is split epic.*

## 5 Von Neumann regularity of elementary CA

▶ **Theorem 14.** *The elementary CA with numbers 6, 7, 23, 33, 57 and 77 are regular.*

**Proof.** It is a finite case analysis to verify that the CA defined in Figure 1, Figure 2, Figure 3, Figure 4, Figure 5 and Figure 6 in Appendix A are generalized inverses of the respective CA. Code for verifying this and discussion on how such rules were found is included in the arXiv version [8]. ◀

▶ **Theorem 15.** *The elementary CA with numbers 9, 27, 28, 41 and 58 are not regular.*

**Proof.** See the lemmas below. ◀

▶ **Lemma 16.** *The elementary CA 9 is not regular.*

**Proof.** Let $f$ be the ECA 9, i.e. $f(x)_i = 1 \iff x_{[i-1,i+1]} \in \{000, 011\}$. The image $X$ of $f$ is the SFT with forbidden patterns 1011, 10101, 11001, 11000011 and 110000101. One can verify[3] this with standard automata-theoretic methods.

We have $f(0^{\mathbb{Z}}) = 1^{\mathbb{Z}}$ and $f(1^{\mathbb{Z}}) = 0^{\mathbb{Z}}$, so if $g : X \to \{0, 1\}^{\mathbb{Z}}$ is a right inverse for $f$, then $g(0^{\mathbb{Z}}) = 1^{\mathbb{Z}}$. Consider now the configuration

$$x = ...0000011.00000... \in X$$

where coordinate 0 is to the left of the decimal point (i.e. the rightmost 1 or the word 11). Let $g(x) = y$. Then $y_i = 1$ for all large enough $i$ and $y_i = 0$ for some $i$. Let $n$ be maximal such that $y_n = 0$. Then $y_{[n,n+2]} = 011$ so $f(y)_{n+1} = 1$ and $f(y)_{n+1+i} = 0$ for all $i \geq 1$. Since $f(y) = x$, we must have $n = -1$ and since $\{000, 011\}$ does not contain a word of the form $a01$, it follows that $f(y)_{-1} = 0 \neq x_{-1}$, a contradiction. ◀

[3] For verifying only the proof of this lemma, i.e. the non-regularity of ECA 9, it is enough to show that the point $x$ below is in $X$, that is, it has some preimage (...0100100001001001... is one). Knowing the SFT is, however, essential for finding such an argument, so we argue in this way, again to illustrate the method.

The proof shows that the CA does not have the strong periodic point property for $p = 1$. In general, for fixed $p$ one can use automata-theory to decide whether it holds up to that $p$, though here (and in all other proofs) we found the contradictions by hand before we had to worry about actually implementing this.

▶ **Lemma 17.** *The ECA* 27 *is not regular.*

**Proof.** Let $f$ be the ECA 27, i.e. $f(x)_i = 1 \iff x_{[i-1,i+1]} \in \{000, 001, 011, 100\}$. The image $X$ of $f$ is proper sofic, we omit the automaton and argue directly in terms of configurations. Proposition 10 directly shows that the CA can not be regular in the case when the image is proper sofic, but we give a direct proof to illustrate the method (and so that we do not have to provide a proof that the image is sofic, which is straightforward but lengthy).

Again, we will see that this CA does not satisfy the strong periodic point condition for $p = 1$. Observe that $f(1^{\mathbb{Z}}) = 0^{\mathbb{Z}}$ and $f(0^{\mathbb{Z}}) = 1^{\mathbb{Z}}$ so if $g$ is a right inverse from the image to $\{0, 1\}^{\mathbb{Z}}$, then $g(0^{\mathbb{Z}}) = 1^{\mathbb{Z}}$ and $g(1^{\mathbb{Z}}) = 0^{\mathbb{Z}}$. Let $y = ...000001100.10101010...$ and observe that

$$f(y) = f(...000001100.10101010...) = \\ ...111111011.00000000... = x \in X.$$

We now reason similarly as in Lemma 16. We have $g(x)_i = 1$ for all large enough $i$, and if $n$ is maximal such that $g(x)_n = 0$, then $f(g(x))_{n+1} = 1$ and $f(g(x))_{n+1+i} = 0$ for all $i \geq 1$, so again necessarily $n = -1$. A short combinatorial analysis shows that no continuation to the left from $n$ produces $f(g(x))_n = 1$ and $f(g(x))_{n-1} = 0$, that is, the image of $g$ has no possible continuation up to coordinate $-1$. ◀

▶ **Lemma 18.** *The ECA* 28 *is not regular.*

**Proof.** Let $f$ be the ECA 28, i.e. $f(x)_i = 1 \iff x_{[i-1,i+1]} \in \{010, 011, 100\}$. The image $X$ of $f$ is the SFT with the single forbidden pattern 111.

We have $f(0^{\mathbb{Z}}) = f(1^{\mathbb{Z}}) = 0^{\mathbb{Z}}$. The point

$$...0000.10000... \in X$$

contradicts the choice $g(0^{\mathbb{Z}}) = 0^{\mathbb{Z}}$ by a similar analysis as in previous theorems; similarly as in Example 6, computing the preimage from right to left, the asymptotic type necessarily changes to 1s. Thus we must have $g(0^{\mathbb{Z}}) = 1^{\mathbb{Z}}$.

On the other hand, if $g(0^{\mathbb{Z}}) = 1^{\mathbb{Z}}$, then going from right to left, we cannot find a preimage for

$$...0001.10000... \in X.$$

(Alternatively, going from left to right, the asymptotic type necessarily changes to 0s or never becomes 1-periodic.)

It follows that $g(0^{\mathbb{Z}})$ has no consistent possible choice, a contradiction. ◀

▶ **Lemma 19.** *The ECA* 41 *is not regular.*

**Proof.** Let $f$ be the ECA 41, i.e. $f(x)_i = 1 \iff x_{[i-1,i+1]} \in \{000, 011, 101\}$. The image $X$ of $f$ is proper sofic, we omit the automaton and argue directly in terms of configurations. Again Proposition 10 would also yield the result.

Again, we will see that this CA does not satisfy the strong periodic point condition for $p = 1$. Observe that $f(1^{\mathbb{Z}}) = 0^{\mathbb{Z}}$ and $f(0^{\mathbb{Z}}) = 1^{\mathbb{Z}}$ so if $g$ is a right inverse from the image to $\{0,1\}^{\mathbb{Z}}$, then $g(0^{\mathbb{Z}}) = 1^{\mathbb{Z}}$ and $g(1^{\mathbb{Z}}) = 0^{\mathbb{Z}}$. Let $y = ...00000001.00100100...$ so

$$f(y) = f(...00000001.00100100...) =$$
$$...11111100.00000000... = x \in X.$$

In the usual way (right to left), we verify that $x$ has no preimage that is right asymptotic to $1^{\mathbb{Z}}$, obtaining a contradiction. ◀

▶ **Lemma 20.** *The ECA* 58 *is not regular.*

**Proof.** Let $f$ be the ECA 58, i.e. $f(x)_i = 1 \iff x_{[i-1,i+1]} \in \{001, 011, 100, 101\}$. The image $X$ of $f$ is proper sofic, we omit the automaton. Again Proposition 10 would also yield the result.

The point $0^{\mathbb{Z}}$ has two 1-periodic preimages. We show neither choice satisfies the strong periodic point condition: if $g(0^{\mathbb{Z}}) = 1^{\mathbb{Z}}$, then $g$ cannot give a preimage for

...0000000.10000000...

If $g(0^{\mathbb{Z}}) = 0^{\mathbb{Z}}$, then it cannot give a preimage for

...0000000.11000000...

It is easy to find preimages for these two configurations, however, so ECA 58 is not regular. ◀

## References


1. Alonso Castillo-Ramirez and Maximilien Gadouleau. Elementary, finite and linear vN-regular cellular automata. *Inform. and Comput.*, 274:104533, 12, 2020. doi:10.1016/j.ic.2020.104533.
2. T. Ceccherini-Silberstein and M. Coornaert. *Cellular Automata and Groups*. Springer Monographs in Mathematics. Springer-Verlag Berlin Heidelberg, 2010. URL: https://books.google.cl/books?id=N-LSFFaHTKwC.
3. Walter Gottschalk. Some general dynamical notions. In *Recent advances in topological dynamics (Proc. Conf. Topological Dynamics, Yale Univ., New Haven, Conn., 1972; in honor of Gustav Arnold Hedlund)*, pages 120–125. Lecture Notes in Math., Vol. 318, 1973.
4. John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation (3rd Edition)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2006.
5. Jarkko Kari. Rice's theorem for the limit sets of cellular automata. *Theoret. Comput. Sci.*, 127(2):229–254, 1994. doi:10.1016/0304-3975(94)90041-8.
6. Douglas Lind and Brian Marcus. *An introduction to symbolic dynamics and coding*. Cambridge University Press, Cambridge, 1995. doi:10.1017/CBO9780511626302.
7. M. Lothaire. *Algebraic combinatorics on words*, volume 90 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, Cambridge, 2002.
8. V. Salo. Von Neumann regularity, split epicness and elementary cellular automata. *ArXiv e-prints*, April 2018. arXiv:1804.03913.
9. Ville Salo and Ilkka Törmä. Category theory of symbolic dynamics. *Theor. Comput. Sci.*, 567:21–45, 2015. doi:10.1016/j.tcs.2014.10.023.
10. Benjamin Weiss. Sofic groups and dynamical systems. *Sankhyā: The Indian Journal of Statistics, Series A (1961-2002)*, 62(3):350–359, 2000. URL: http://www.jstor.org/stable/25051326.
11. Stephen Wolfram. Statistical mechanics of cellular automata. *Rev. Modern Phys.*, 55(3):601–644, 1983. doi:10.1103/RevModPhys.55.601.

## A Weak generalized inverses

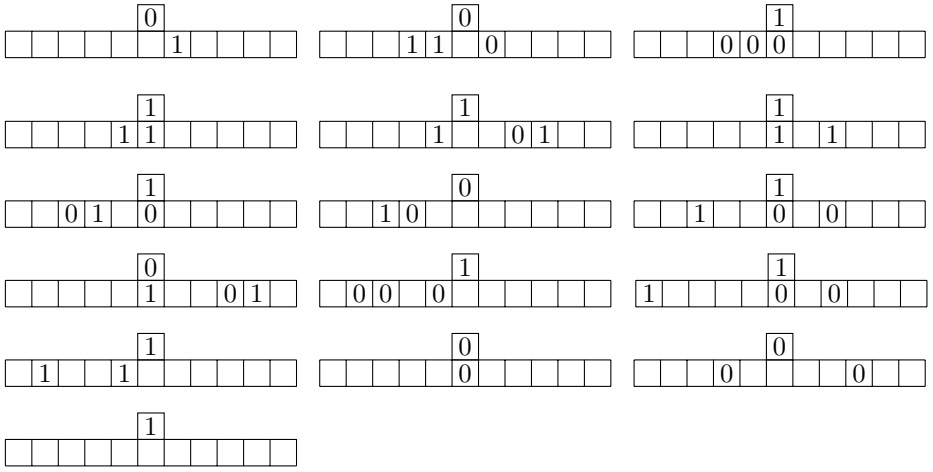


**Figure 1** A weak generalized inverse of ECA 6. The rules are applied row by row, and on each row from left to right. An empty box denotes a wildcard symbol, and the first rule to apply is used. The rightmost coordinate is not actually read by any rule. This is the radius 5 binary CA with the hex number

```
00000000000000000000FFF3000000FF0000FFFF000000F00000FF000000FFFF
00000000000000000000FFF30000FFFF0000FFFF0000FFFF0000FFF00000FFFF
00000000000000000000FFF3000000FF0000FFFF000000F00000FF000000FFFF
00000000000000000000FFF30000FFFF0000FFFF000000FF0000FFF30000FFFF
00000000000000000000FFF3000000FF0000FFFF000000F00000FF000000FFFF
00000000000000000000FFF30000FFFF0000FFFF0000FFFF0000FFF00000FFFF
00000000000000000000FFF3000000FF0000FFFF000000F00000FF000000FFFF
00000000000000000000FFF30000FFFF0000FFFF000000F00000FFF30000FFFF
```

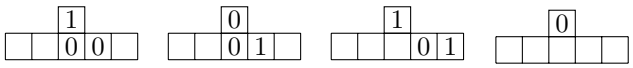


**Figure 2** A weak generalized inverse of ECA 7. This is the radius 2 binary CA with the hex number 23232323, equal to ECA 35 composed with $\sigma$.

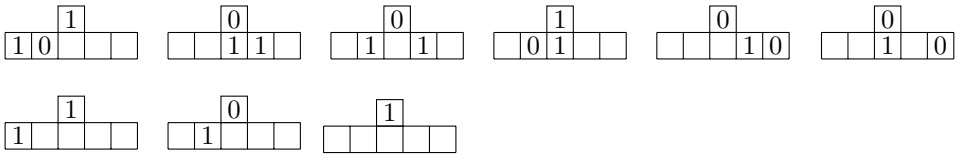


**Figure 3** A weak generalized inverse of ECA 23. This is the radius 2 binary CA with the hex number 23FF003B.

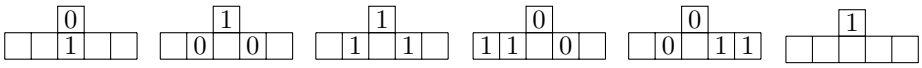


**Figure 4** A weak generalized inverse of ECA 33. This is the radius 2 binary CA with the hex number `0C070F07`.

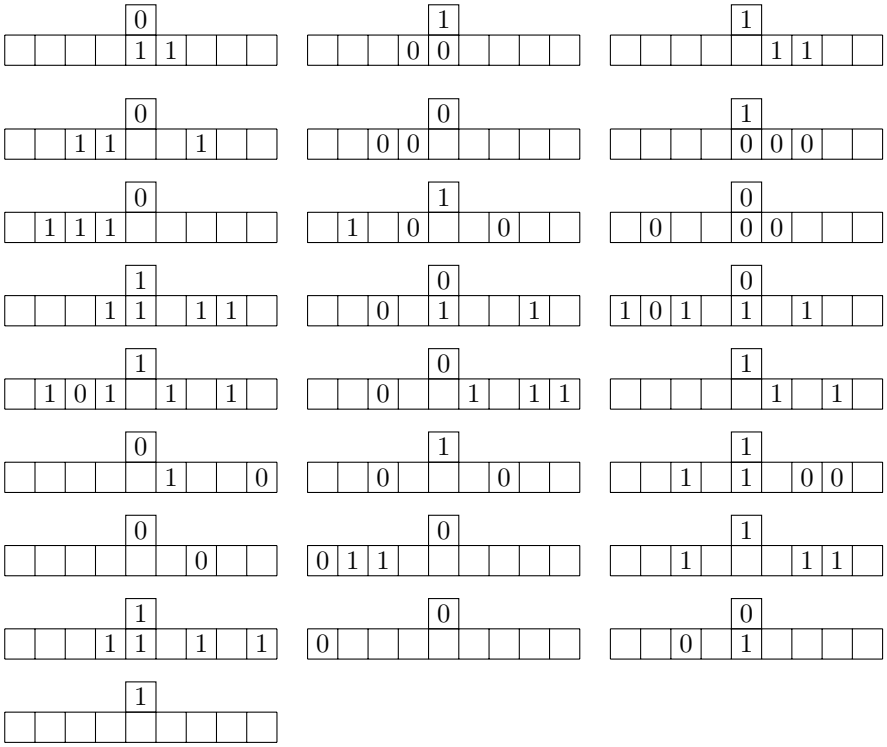


**Figure 5** A weak generalized inverse of ECA 57. This is the radius 4 binary cellular automaton with the hex number

`0000F00F00FFFFFF00E3FEFF0000FFFF0003FC0F0003FFFF00E3F60F0000FFFF`
`0000F00F000FFFFF00E3FE0F0000FFFF0003FC0F00C3FFFF00E3F60F0000FFFF`

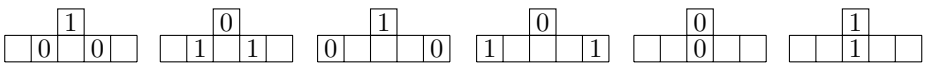


**Figure 6** A weak generalized inverse of ECA 77. This is the radius 2 binary CA with the hex number `107331F7`.

# State-Based Opacity of Real-Time Automata

**Kuize Zhang** ✉
Control Systems Group, Technische Universität Berlin, Germany

**Abstract**

State-based opacity is a special type of opacity as a confidentiality property, which describes whether an external intruder cannot make for sure whether secret states of a system have been visited by observing generated outputs, given that the intruder knows complete knowledge of the system's structure but can only see generated outputs. When the time of visiting secret states is specified as the initial time, the current time, any past time, and at most $K$ steps prior to the current time, the notions of state-based opacity can be formulated as initial-state opacity, current-state opacity, infinite-step opacity, and $K$-step opacity, respectively. In this paper, we formulate the four versions of opacity for real-time automata which are a widely-used model of real-time systems, and give 2-EXPTIME verification algorithms for the four notions by defining appropriate notions of observer and reverse observer for real-time automata that are computable in 2-EXPTIME.

**2012 ACM Subject Classification** Theory of computation → Timed and hybrid models; Security and privacy → Formal security models

**Keywords and phrases** real-time automaton, state-based opacity, observer, verification

**Digital Object Identifier** 10.4230/OASIcs.AUTOMATA.2021.12

**Funding** This work was partially supported by the Alexander von Humboldt Foundation.

## 1 Introduction

### 1.1 Background

*Opacity* is a confidentiality property that is firstly proposed in [8] to characterize information flow security, and has been widely used to describe all kinds of scenarios in security/privacy problems. It describes whether a system can forbid an external intruder from making for sure whether some secrets have been visited by using observed outputs, given that the intruder knows complete knowledge of the system's structure but can only see outputs generated by the system. In [3], a general *run-based opacity* framework is proposed for labeled transition systems (LTSs), where such a system is opaque if for every secret run, there exists a non-secret run such that the two runs have the same observation. Later on, two special types of secrets are studied: subsets of event sequences (aka traces) and subsets of states. According to the two types of secrets, opacity is classified into *language-based opacity* and *state-based opacity*. The former refers to as for every generated secret trace, there is a non-secret generated trace such that they have the same observation; the latter means whenever a run passes through a secret state at some instant, there exists another run that does not pass any secret state at the same instant such that the two runs have the same observation.

### 1.2 Literature review

In order to apply opacity to different scenarios, different notions of opacity in different models have been studied, e.g., four types of state-based opacity, called *initial-state opacity* (ISO) [13], *current-state opacity* (CSO) [5], *infinite-step opacity* (InfSO) [12], and *$K$-step opacity* (KSO) [11] for labeled finite automata (LFAs) are proved to be decidable in PSPACE with PSPACE lower bounds in [13, 5, 12] and with an NP lower bound in [11]. Unlike LFAs, ISO of labeled Petri nets is undecidable [3, 16]. Language-based opacity is more involved, because it is already undecidable in finite LTSs (i.e., LFAs) with $\epsilon$-labeling functions [3]. In [7],


language-based opacity is shown to be decidable in EXPTIME for LFAs when secret languages and non-secret languages are regular. Moreover, in [19, 1], ISO, CSO, InfSO, KSO, and the special language-based opacity (as in [7]) in LFAs are reduced to each other all in polynomial time, so the decision problems for the five definitions of opacity are all PSPACE-complete. Other related opacity results for untimed systems can be found in [19, 20, 2, 6], etc.

In contrast to untimed systems, in real-time systems, except for an observable transition, the execution of an unobservable transition may also cost time, so the study of opacity in real-time systems is much more complicated, and there have been rare opacity results. In [4], a notion of L-opacity (the counterpart of CSO generalized to timed automata) is proved to be undecidable for a very restrictive class of timed automata called event recording automata, in which each clock is associated with an event and when an event occurs the corresponding clock is reset. Later in [18, 17], language-based opacity of *real-time automata* (RTAs, in which there is a single clock that is reset at the occurrence of each event) is proved to be decidable, where the secret languages and non-secret languages are those recognized by RTAs, hence ISO is also decidable as a special case of the considered language-based opacity. The overall verification idea is to compute the intersection of the secret language and the complement of the non-secret language for a given RTA.

## 1.3 Contribution of the paper

In this paper, we formulate the above mentioned four types of state-based opacity (ISO, CSO, InfSO, and KSO) for an RTA $\mathcal{A}$ (where the ISO is the same as that in [18, 17]), and show that they are all decidable in 2-EXPTIME. The verification method used in the current paper (totally different from the one used in [18, 17]) is firstly to define and compute notions of *observer* $\mathcal{A}_{\text{obs}}$ and *reverse observer* ${}_{\mathsf{R}}\mathcal{A}_{\text{obs}}$ in 2-EXPTIME in the size of $\mathcal{A}$, and secondly use $\mathcal{A}_{\text{obs}}$ and ${}_{\mathsf{R}}\mathcal{A}_{\text{obs}}$ to verify the four notions of opacity in time linear or quadratic polynomial in the sizes of $\mathcal{A}_{\text{obs}}$ and ${}_{\mathsf{R}}\mathcal{A}_{\text{obs}}$. Compared with LFAs, the transitions of RTAs carry real intervals which denote that the time consumption of a transition's execution may be any real in the corresponding interval, so that RTAs can represent real-time systems. The considerable difficulty in characterizing opacity for RTAs compared with that for LFAs just comes from the intervals in RTAs. Note that the method of using observers to verify opacity is a conventional one used in LFAs, e.g., in [10], a notion of observer (actually the powerset construction used for determinizing nondeterministic finite automata with $\epsilon$-transitions [15]) is used to verify CSO of LFAs; in [19], a notion of reverse observer is used to verify ISO of LFAs; in [20], a notion of two-way observer (combining an observer and a reverse observer) is used to verify InfSO and KSO, all in EXPTIME. The essential technical difficulty of the current paper lies in how to define suitable notions of observer and reverse observer for RTAs and how to compute them.

The remainder is structured as follows. In Section 2, we introduce necessary notation, show the exact run length problem that belongs to NP, and also introduce based knowledge in RTAs; In Section 3, we show the main results of the paper, which include four definitions of state-based opacity for RTAs, notions of observer and reverse observer and how to compute them, and necessary and sufficient conditions for the four definitions of state-based opacity. Section 4 ends up the paper with a short conclusion.

# 2 Preliminaries

## 2.1 Notation

Symbols $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Z}_+$, $\mathbb{Q}$, $\mathbb{Q}_{\geqslant 0}$, $\mathbb{R}$, and $\mathbb{R}_{\geqslant 0}$ denote the sets of nonnegative integers, integers, positive integers, rational numbers, nonnegative rational numbers, real numbers, and nonnegative real numbers, respectively. For $a, b \in \mathbb{R} \cup \{\pm\infty\}$ such that $a \leqslant b$, we use $\langle a, b\rangle$ to denote an interval, where "$\langle$" represents "[" (left-closed) or "(" (left-open), "$\rangle$" represents "]" (right-closed) or ")" (right-open). For a finite alphabet $\Sigma$, $\Sigma^*$ denotes the set of *words* over $\Sigma$ including the empty word $\epsilon$. $\Sigma^+ := \Sigma^* \backslash \{\epsilon\}$. For a word $s = s_1 s_2 \dots s_n \in \Sigma^*$, $|s|$ stands for its *length* $n$, $s^{\mathsf{R}}$ denotes the *mirror image* $s_n \dots s_2 s_1$ of $s$. For $s \in \Sigma^+$ and $k \in \mathbb{N}$, $s^k$ denotes the concatenation of $k$ copies of $s$. For a word $s \in \Sigma^*$, a word $s' \in \Sigma^*$ is called a *prefix* of $s$, denoted as $s' \sqsubset s$, if there exists another word $s'' \in \Sigma^*$ such that $s = s's''$. For $s \in \Sigma^*$ and $s' \sqsubset s$, we use $s \backslash s'$ to denote the word $s''$ such that $s's'' = s$. For two nonnegative integers $i \leqslant j$, $[\![i, j]\!]$ denotes the set of all integers no less than $i$ and no greater than $j$; for a set $S$, $|S|$ denotes its cardinality and $2^S$ its power set.

## 2.2 The exact run length problem

Let **Int** be the set of nonempty intervals of $\mathbb{R}$ having left endpoints in $\mathbb{Q} \cup \{-\infty\}$ and right endpoints in $\mathbb{Q} \cup \{+\infty\}$. Note that $\pm\infty$ do not belong to any interval of **Int**. Consider a $k$-dimensional *duration directed graph* $G = (\mathbf{Int}^k, V, A)$, where $k \in \mathbb{Z}_+$, $\mathbf{Int}^k$ is the $k$-fold Cartesian product of **Int**, $V$ a finite set of vertices, $A \subset V \times \mathbf{Int}^k \times V$ a finite set of directed edges (arcs) with weights in $\mathbf{Int}^k$. A *run* of $G$ is defined by $v_0 \xrightarrow{z_1} v_1 \xrightarrow{z_2} \cdots \xrightarrow{z_n} v_n =: r$, where $n \in \mathbb{Z}_+$, $v_0, \dots, v_n \in V$, for all $i \in [\![1, n]\!]$, $z_i = (z_i(1), \dots, z_i(k)) \in \mathbb{R}^k$, $(v_{i-1}, int_i, v_i) \in A$ for some $int_i = (int_i(1), \dots, int_i(k)) \in \mathbf{Int}^k$, and $z_i(j) \in int_i(j)$ for all $j \in [\![1, k]\!]$. The *weight* of run $r$ is defined by $\sum_{i=1}^n z_i$. We sometimes write $v_1 \to v_2$ to denote a run from $v_1$ to $v_2$ without specifying the intermediate vertices and vectors. For an edge $(v_1, int_{v_1v_2}, v_2) =: a \in A$, we denote $\mathrm{tail}(a) = v_1$ and $\mathrm{head}(a) = v_2$.

▶ **Problem 1** (ERL). *Given a positive integer $k$, a $k$-dimensional duration directed graph $G = (\mathbf{Int}^k, V, A)$, two vertices $v_1, v_2 \in V$, and a vector $z \in \mathbb{Q}^k$, determine whether there exists a run from $v_1$ to $v_2$ with weight $z$.*

We set as usual for $n \in \mathbb{Z}_+$, the size $\mathrm{size}(n)$ of $n$ to be the length of its binary representation; then $\mathrm{size}(-n) = \mathrm{size}(n) + 1$; $\mathrm{size}(0) = 1$; for a rational number $m/n$, where $m, n$ are relatively prime integers, $\mathrm{size}(m/n) = \mathrm{size}(m) + \mathrm{size}(n)$, then for a vector $z \in \mathbb{Q}^k$, its size is the sum of the sizes of its components. In addition, $\mathrm{size}(+\infty) = \mathrm{size}(-\infty) = 2$. For a duration directed graph $G = (\mathbf{Int}^k, V, A)$, for every edge $(v_1, int_{v_1v_2}, v_2) \in A$, denote $int_{v_1v_2} = (int_{v_1v_2}(1), \dots, int_{v_1v_2}(k))$, where $int_{v_1v_2}(i) = \langle a^i_{v_1v_2}, b^i_{v_1v_2}\rangle$, $a^i_{v_1v_2} \in \mathbb{Q} \cup \{-\infty\}$, $b^i_{v_1v_2} \in \mathbb{Q} \cup \{+\infty\}$, $i \in [\![1, k]\!]$. The size $\mathrm{size}(G)$ of a given graph $G$ is equal to $|V| + \mathrm{size}(A) = |V| + \sum_{(v_1, int_{v_1v_2}, v_2) \in A}(2 + 2k + \sum_{i=1}^k (\mathrm{size}(a^i_{v_1v_2}) + \mathrm{size}(b^i_{v_1v_2})))$. Then the size of an instance $(k, G, v_1, v_2, z)$ of the ERL problem is $\mathrm{size}(k) + \mathrm{size}(G) + 2 + \mathrm{size}(z)$.

For every edge $(v_1, int_{v_1v_2}, v_2) \in A$ (sometimes also written as $(v_1, v_2) \in A$ for short), we denote $w^1_{v_1v_2} = (a^1_{v_1v_2}, \dots, a^k_{v_1v_2}) \in (\mathbb{Q} \cup \{-\infty\})^k$ and $w^2_{v_1v_2} = (b^1_{v_1v_2}, \dots, b^k_{v_1v_2}) \in (\mathbb{Q} \cup \{+\infty\})^k$. We set as usual $-\infty < a < +\infty$, $a + (\pm\infty) = (\pm\infty) + a = \pm\infty$ for all $a \in \mathbb{R}$, $b \cdot (\pm\infty) = (\pm\infty) \cdot b = \pm\infty$ for all $0 \neq b \in \mathbb{R}$. We also set $0 \cdot (\pm\infty) = (\pm\infty) \cdot 0 = 0$. For two vectors $z_1, z_2$ in $(\mathbb{Q} \cup \{\pm\infty\})^k$, we write $z_1 \leqslant z_2$ and $z_2 \geqslant z_1$ if $z_1(i) \leqslant z_2(i)$ for all $i \in [\![1, k]\!]$. For $r \in \mathbb{R}$ and $z = (z(1), \dots, z(k)) \in \mathbb{R}^k$, we write $r + z = z + r = (z(1) + r, \dots, z(k) + r)$.

▶ **Lemma 1.** *The ERL problem belongs to* NP.

**Proof.** Consider an instance $(k, G, v_1, v_2, z)$ of the ERL problem.

We add an edge $(v_2, \underbrace{([0,0], \ldots, [0,0])}_{k}, v_1) =: \bar{a}$, then $w^1_{\bar{a}} = w^2_{\bar{a}} = \underbrace{(0, \ldots, 0)}_{k} =: 0^k$. For each edge $a$ in $A$, we define a variable $x_a$. For edge $\bar{a}$, we also define a variable $x_{\bar{a}}$. Consider the following inequality

$$x_{\bar{a}} w^1_{\bar{a}} + \sum_{a \in A} x_a w^1_a \leqslant z, \tag{1a}$$

$$-x_{\bar{a}} w^2_{\bar{a}} - \sum_{a \in A} x_a w^2_a \leqslant -z, \tag{1b}$$

with constraints

$$x_{\bar{a}} = 1, \tag{2a}$$

$$x_a \in \mathbb{N} \text{ for all } a \in A, \tag{2b}$$

$$\sum_{\substack{a \in A \cup \{\bar{a}\} \\ \text{head}(a) = v}} x_a = \sum_{\substack{a \in A \cup \{\bar{a}\} \\ \text{tail}(a) = v}} x_a \text{ for all } v \in V, \tag{2c}$$

$$\text{the edges } a \text{ such that } x_a > 0 \text{ form a strongly connected component,} \tag{2d}$$

$$\text{for every } a \in A, \text{ if } x_a > 0, \text{ then for all } i \in [\![1, k]\!], \text{ if } \langle a^i_a, b^i_a \rangle \text{ is left-open (resp., right-open), then the } i\text{-th component of (1a) (resp., (1b)) must hold strictly.} \tag{2e}$$

If $(\tilde{x}_a)_{a \in A \cup \{\bar{a}\}}$ is a solution to (1) satisfying constrains (2), then there exists a run from $v_1$ to $v_2$ having $\tilde{x}_a$ repetitive edges $a \in A$ therein with weight $z$ by continuity of $\mathbb{R}$, i.e., $(k, G, v_1, v_2, z)$ is a positive instance of the ERL problem. If (1) has no solution satisfying constraints (2), then $(k, G, v_1, v_2, z)$ is a negative instance of the ERL problem.

For an edge $a \in A$ and $i \in [\![1, k]\!]$ such that $w^1_a(i) = -\infty$, we either put $x_a = 0$ into (1) (in this case, $w^1_a(i)$ is eliminated) or remove the $i$-th component from (1a) in case $x_a \geqslant 1$ (in this case, $x_a \geqslant 1$ implies that the $i$-th component of (1a) always holds no matter what values the other variables are in). For an edge $a \in A$ and $i \in [\![1, k]\!]$ such that $w^2_a(i) = +\infty$, we do similar things. Then we obtain a number $\leqslant 4^{k|A|}$ of standard integer linear programming [14, Cor. 17.1d] (i.e., of the form $Ax \leqslant b$ with constraints, where constants $A \in \mathbb{Q}^{m \times n}$, $b \in \mathbb{Q}^{m \times 1}$, variables $x \in \mathbb{N}^{n \times 1}$) that is solvable in NP. Hence inequality (1) with constraints in (2) can be solved in NP in the size of the instance $(k, G, v_1, v_2, z)$. ◀

▶ Remark 2. For a duration directed graph $G$, if all intervals shrink to a singleton, then the ERL problem reduces to the NP-complete *exact path length* problem proved in [9]. The proof of Lemma 1 is inspired by the proof of the NP membership of the exact path length problem in [9] but is more involved.

In order to obtain our main results on verification of state-based opacity for RTAs, we need the following *component-length-equal run* (CLER) problem.

▶ **Problem 2** (CLER). *Given a positive integer $k > 1$, a $k$-dimensional duration directed graph $G = (\mathbf{Int}^k, V, A)$, two vertices $v_1, v_2 \in V$, and an edge $(v_2, int_{v_2v_3}, v_3) \in A$, determine whether there exists a run $v_1 \to v_2 \xrightarrow{w_{v_2v_3}} v_3$ whose weight has equal components in $\mathbb{Q}$, where $v_1 \to v_2$ does not contain $v_3$, $w_{v_2v_3}(i) \in int_{v_2v_3}(i)$ for all $i \in [\![1, k]\!]$.*

▶ **Lemma 3.** *The CLER problem belongs to* NP.

**Proof.** Consider a $k$-dimensional ($k > 1$) duration directed graph $G = (\mathbf{Int}^k, V, A)$, and an instance $(k, G, v_1, v_2, v_3)$ of the CLER problem, where $(v_2, int_{v_2v_3}, v_3) \in A$ for some $int_{v_2v_3} \in \mathbf{Int}^k$. We next construct a $(k-1)$-dimensional duration directed graph $G' = (\mathbf{Int}^{k-1}, V, A')$, and transform the instance $(k, G, v_1, v_2, v_3)$ of the CLER problem to an instance $(k-1, G', v_1, v_3, 0^{k-1})$ of the ERL problem. $G'$ is obtained from $G$ as follows: for every edge $(v', int_{v'v''}, v'') \in A$, denote $int_{v'v''} = (\langle a^1_{v'v''}, b^1_{v'v''}\rangle, \ldots, \langle a^k_{v'v''}, b^k_{v'v''}\rangle) \in \mathbf{Int}^k$, we compute $int'_{v'v''} = (\langle a^1_{v'v''} - b^k_{v'v''}, b^1_{v'v''} - a^k_{v'v''}\rangle, \ldots, \langle a^{k-1}_{v'v''} - b^k_{v'v''}, b^{k-1}_{v'v''} - a^k_{v'v''}\rangle) \in \mathbf{Int}^{k-1}$, where for all $i \in [\![1, k-1]\!]$, $\langle a^i_{v'v''} - b^k_{v'v''}, b^i_{v'v''} - a^k_{v'v''}\rangle$ is left-open (resp., right-open) if and only if either $\langle a^i_{v'v''}, b^i_{v'v''}\rangle$ is left-open or $\langle a^k_{v'v''}, b^k_{v'v''}\rangle$ is right-open (resp., either $\langle a^i_{v'v''}, b^i_{v'v''}\rangle$ is right-open or $\langle a^k_{v'v''}, b^k_{v'v''}\rangle$ is left-open). We set $A' = \{(v', int'_{v'v''}, v'') | (v', int_{v'v''}, v'') \in A\}$.

We then have (i) $(k, G, v_1, v_2, v_3)$ is a positive instance of the CLER problem if and only if (ii) $(k-1, G', v_1, v_3, 0^{k-1})$ is an instance of the ERL problem such that in $G'$, there exists a run $v_1 \to v_2 \xrightarrow{w'_{v_2v_3}} v_3$ with weight $0^{k-1}$, where $w'_{v_2v_3}(i) \in int'_{v_2v_3}(i)$ for all $i \in [\![1, k-1]\!]$ and $v_3$ appears only once in the run. Note that (ii) implies that there exists a run $v_1 \to v_2 \xrightarrow{w_{v_2v_3}} v_3$ in $G$ whose weight has equal components $w$ in $\mathbb{R}$, where $v_1 \to v_2$ does not contain $v_3$, $w_{v_2v_3}(i) \in int_{v_2v_3}(i)$ for all $i \in [\![1, k]\!]$. If $w$ is irrational, we can add a very small real number $\varepsilon$ to all components of $w_{v_2v_3}$ such that $v_2 \xrightarrow{w_{v_2v_3}+\varepsilon} v_3$ is still a run and $w + \varepsilon \in \mathbb{Q}$, because $\mathbb{Q}$ is dense in $\mathbb{R}$, hence (i) holds. In order to check whether $(k-1, G', v_1, v_3, 0^{k-1})$ meets the satisfaction, by Lemma 1, we need to add the additional constraint $x_{(v_2,v_3)} = 1$ into the corresponding constraints (2), and then solve the corresponding inequality (1) with the modified constraints. Hence the CLER problem belongs to NP. ◀

## 2.3 Real-time automata

A *real-time automaton* (RTA) is a tuple $\mathcal{A} = (Q, E, Q_0, \Delta, \mu, \Sigma, \ell)$, where $Q$ is a nonempty finite set of *states*, $E$ a finite *alphabet* (elements of $E$ are called *events*), $Q_0 \subset Q$ a nonempty set of *initial states*, $\Delta \subset Q \times E \times Q$ a *transition relation* (elements of $\Delta$ are called *transitions*), $\mu$ assigns to each transition $(q, e, q') \in \Delta$ (also written as $q \xrightarrow{e} q'$) a nonempty interval $\mu(e)_{qq'}$ of $\mathbb{R}_{\geqslant 0}$ with left endpoint and right endpoint being $a$ and $b$, where $a \in \mathbb{Q}_{\geqslant 0}$, $b \in \mathbb{Q}_{\geqslant 0} \cup \{+\infty\}$[1], $a \leqslant b$, $\Sigma$ is a finite set of *outputs*, and $\ell : E \to \Sigma \cup \{\epsilon\}$ is an *output/labeling function.* A state $q \in Q$ is called *dead* if for all $e \in E$ and $q' \in Q$, $(q, e, q') \notin \Delta$.

The size of a given $\mathcal{A}$ is defined by $|Q| + |Q_0| + |\Delta| + \text{size}(\mu) + \text{size}(\ell)$, where the sizes of $\pm\infty$ and rational numbers have been defined before, $\text{size}(\mu) = \sum_{(q,e,q')\in\Delta} \text{size}(\mu(e)_{qq'}) = \sum_{(q,e,q')\in\Delta}(2 + \text{size}(a_{q,e,q'}) + \text{size}(b_{q,e,q'}))$, $a_{q,e,q'}$ and $b_{q,e,q'}$ are the endpoints of the interval $\mu(e)_{qq'}$, 2 is used to denote the sum of the sizes of "[" (resp., "(") and "]" (resp., ")"), $\text{size}(\ell) = |\{(e, \ell(e)) | e \in E\}|$.

A transition $(q, e, q') \in \Delta$ is interpreted as when $\mathcal{A}$ is in state $q$ and event $e$ occurs after some time segment in $\mu(e)_{qq'}$, $\mathcal{A}$ transitions to state $q'$. When event $e \in E$ occurs, the *output* $\ell(e)$ of $e$ will be observed if $\ell(e) \neq \epsilon$ (in this case we call $e$ *observable*); while nothing will be observed if $\ell(e) = \epsilon$ (in this case we call $e$ *unobservable*). A transition $(q, e, q')$ is called *observable (resp., unobservable)* if $e$ is observable (resp., unobservable). We denote by $E_o$ and $E_{uo}$ the sets of observable events and unobservable events, respectively. Output function $\ell$ is extended to $E \times \mathbb{R}_{\geqslant 0}$ as follows: $\ell((e, t)) = (\ell(e), t)$ if $e \in E_o$, $\ell((e, t)) = \epsilon$ otherwise. Then $\ell$ is recursively extended to $E^*$ as $\ell(e_1 \ldots e_n) = \ell(e_1) \ldots \ell(e_n)$ and also to $(E \times \mathbb{R}_{\geqslant 0})^*$ analogously.

[1] When $b = +\infty$, the possible intervals can only be of the form $[a, +\infty)$ or $(a, +\infty)$; when $a = b$, the possible intervals can only be $[a, a] = \{a\}$.

A *path* of $\mathcal{A}$ is defined by the empty word $\epsilon$ or a sequence $q_0 \xrightarrow{e_1} q_1 \xrightarrow{e_2} \cdots \xrightarrow{e_n} q_n$, where $n \in \mathbb{Z}_+$, $(q_{i-1}, e_i, q_i) \in \Delta$ for all $i \in [\![1, n]\!]$. A path is called a *cycle* if its start state and terminal state coincide. A *run* of $\mathcal{A}$ is either $\epsilon$ or a sequence $q_0 \xrightarrow{e_1/t_1} q_1 \xrightarrow{e_2/t_2} \cdots \xrightarrow{e_n/t_n} q_n =: \pi$, where, $n \in \mathbb{Z}_+$, $(q_{i-1}, e_i, q_i) \in \Delta$, $t_i \in \mu(e_i)_{q_{i-1}q_i}$ for all $i \in [\![1, n]\!]$. The *timed word* of run $\pi$ is defined by $\tau(\pi) = (e_1, t'_1)(e_2, t'_2)\dots(e_n, t'_n)$, where $t'_i = \sum_{k=1}^{i} t_k$ for all $i \in [\![1, n]\!]$. The *weight* $\mathrm{WT}_\pi$ of $\pi$ is defined by $t'_n$. A run is called *instantaneous* if its weight is equal to 0, and called *noninstantaneous* otherwise. A run $\pi$ is called *unobservable* if $\ell(e_1 \dots e_n) = \epsilon$, and called *observable* otherwise. A path can also be defined to be either unobservable or observable analogously. We use $\mathrm{init}(\pi)$ (resp., $\mathrm{last}(\pi)$) to denote its first state $q_0$ (resp., last state $q_n$), respectively. For a set $\Pi$ of runs, we use $\mathrm{init}(\Pi)$ (resp., $\mathrm{last}(\Pi)$) to denote the set of the initial states (resp., last states) of the runs of $\Pi$. The set of runs starting at $q_0 \in Q$ and ending at $q \in Q$ is denoted by $q_0 \rightsquigarrow q$. For $e_1, \dots, e_n \in E$, $q_0 \overset{e_1 \dots e_n}{\rightsquigarrow} q$ denotes the set of all runs of the form $q_0 \xrightarrow{e_1/t_1} q_1 \xrightarrow{e_2/t_2} \cdots \xrightarrow{e_{n-1}/t_{n-1}} q_{n-1} \xrightarrow{e_n/t_n} q$, where $q_1, \dots, q_{n-1} \in Q$. For two runs $\pi_1 = q_0 \xrightarrow{e_1/t_1} q_1 \xrightarrow{e_2/t_2} \cdots \xrightarrow{e_n/t_n} q_n$ and $\pi_2 = q_n \xrightarrow{e_{n+1}/t_{n+1}} q_{n+1} \xrightarrow{e_{n+2}/t_{n+2}} \cdots \xrightarrow{e_{n+m}/t_{n+m}} q_{n+m}$, we use $\pi_1\pi_2$ to denote the concatenation $q_0 \xrightarrow{e_1/t_1} q_1 \xrightarrow{e_2/t_2} \cdots \xrightarrow{e_{n+m}/t_{n+m}} q_{n+m}$ (removing either $\mathrm{last}(\pi_1)$ or $\mathrm{init}(\pi_2)$). For a run $\pi$ satisfying $q_0 \in Q_0$, $\ell(\tau(\pi)) \in (\Sigma \times \mathbb{R}_{\geqslant 0})^*$ is called a *timed output sequence generated by* $\mathcal{A}$. In this case, we observe $\ell(e_i)$ at time $t'_i$ if $e_i \in E_o$, observe nothing at time $t'_i$ if $e_i \in E_{uo}$, $i \in [\![1, n]\!]$. We extend function $\tau$ as follows: for all $\gamma = (\sigma_1, t_1)\dots(\sigma_n, t_n) \in (\Sigma \times \mathbb{R}_{\geqslant 0})^*$,

$$\tau(\gamma) = (\sigma_1, t'_1)\dots(\sigma_n, t'_n), \tag{3}$$

where $t'_j = \sum_{i=1}^{j} t_i$ for all $j \in [\![1, n]\!]$. The *timed language* $L(\mathcal{A})$ generated by $\mathcal{A}$ is denoted by the set of timed words of all runs of $\mathcal{A}$ starting from initial states; $\mathcal{L}(\mathcal{A})$ is the set of timed output sequences generated by $\mathcal{A}$. For a sequence $\gamma \in (\Sigma \times \mathbb{R}_{\geqslant 0})^*$, we use $[\gamma]$ to denote the set of runs $\pi$ of $\mathcal{A}$ starting from initial states such that $\ell(\tau(\pi)) = \gamma$. For $w = (e_1, t_1)(e_2, t_2)\dots(e_n, t_n) \in (\Sigma \times \mathbb{R})^*$ and $t \in \mathbb{R}$, we define $w \pm t = t \pm w = (e_1, t_1 \pm t)(e_2, t_2 \pm t)\dots(e_n, t_n \pm t)$; define $\mathrm{init}(w) = (\mathrm{init}_L(w), \mathrm{init}_R(w)) = (e_1, t_1)$, $\mathrm{last}(w) = (\mathrm{last}_L(w), \mathrm{last}_R(w)) = (e_n, t_n)$; and also define $\tau^{-1}(w) = (e_1, t_1)(e_2, t_2 - t_1)\dots(e_n, t_n - t_{n-1})$. Hence for a run $q_0 \xrightarrow{e_1/t_1} q_1 \xrightarrow{e_2/t_2} \cdots \xrightarrow{e_n/t_n} q_n =: \pi$, $(\tau^{-1} \circ \tau)(\pi) = (e_1, t_1)(e_2, t_2)\dots(e_n, t_n)$. For $\gamma_1\gamma_2 \in \mathcal{L}(\mathcal{A})$, we use $\mathrm{interm}(\gamma_1, \gamma_2) = \{q \in Q | (\exists \text{ runs } \pi_1, \pi_2)[(\mathrm{init}(\pi_1) \in Q_0) \wedge (\mathrm{last}(\pi_1) = \mathrm{init}(\pi_2) = q) \wedge (\ell(\tau(\pi_1)) = \gamma_1) \wedge (\ell(\tau(\pi_1\pi_2)) = \gamma_1\gamma_2) \wedge (\mathrm{WT}_{\pi_1} = \mathrm{last}_R(\gamma_1)) \wedge (\mathrm{WT}_{\pi_2} = \mathrm{last}_R(\gamma_2) - \mathrm{last}_R(\gamma_1))]\}$ to denote the set of states $\mathcal{A}$ can be in when $\mathcal{A}$ has just generated timed output sequence $\gamma_1$, given that the current observation is timed output sequence $\gamma_1\gamma_2$.

# 3 Main results

## 3.1 Current-state estimate

For $\mathcal{A}$, a subset $x \subset Q$ of states, and a sequence $\gamma \in (\Sigma \times \mathbb{R}_{\geqslant 0})^+$, we define the *current-state estimate* as

$$\begin{aligned}\mathcal{M}(\mathcal{A}, \gamma | x) := \{q \in Q | (\exists q_0 \in x)(\exists n \in \mathbb{Z}_+)(\exists m \in \mathbb{N}) \\ \left(\exists \text{ a run } \pi = q_0 \xrightarrow{e_1/t_1} \cdots \xrightarrow{e_n/t_n} q_n \xrightarrow{e_{n+1}/0} \cdots \xrightarrow{e_{n+m}/0} q\right) \\ [(e_n \in E_o) \wedge (e_{n+1} \dots e_{n+m} \in (E_{uo})^*) \wedge \ell(\tau(\pi)) = \gamma]\}.\end{aligned} \tag{4}$$

Particularly for $\mathcal{A}$ and $x \subset Q$, we define the *instantaneous-state estimate* as

$$\mathcal{M}(\mathcal{A}, \epsilon|x) := x \cup \{q \in Q | (\exists q_0 \in x)(\exists n \in \mathbb{Z}_+)(\exists \text{ a run } \pi = q_0 \xrightarrow{e_1/0} \cdots \xrightarrow{e_n/0} q) \\ [e_1 \dots e_n \in (E_{uo})^*]\}. \tag{5}$$

For all $\gamma \in (\Sigma \times \mathbb{R}_{\geqslant 0})^*$, $\mathcal{M}(\mathcal{A}, \gamma|Q_0)$ is also rewritten as $\mathcal{M}(\mathcal{A}, \gamma)$ for short. Intuitively, for $\gamma = (\sigma_1, t_1) \dots (\sigma_n, t_n) \in (\Sigma \times \mathbb{R}_{\geqslant 0})^+$, $\mathcal{M}(\mathcal{A}, \gamma|x)$ denotes the set of states $\mathcal{A}$ can be in when $\gamma$ has just been generated by $\mathcal{A}$ since $\mathcal{A}$ started from some state of $x$. Hence $\mathcal{M}(\mathcal{A}, \gamma) \subset \text{last}([\gamma])$, and $\subsetneq$ may hold. In order to fit the setting of current-state estimate, after the occurrence of the last observable event $e_n$ (i.e., $e_n$ occurs at the current time), we only allow unobservable, instantaneous runs, which is represented by $q_n \xrightarrow{e_{n+1}/0} \cdots \xrightarrow{e_{n+m}/0} q$ and $e_{n+1} \dots e_{n+m} \in (E_{uo})^*$. Particularly, $\mathcal{M}(\mathcal{A}, \epsilon|x)$ denotes the set of states $\mathcal{A}$ can be in at the instant when $\mathcal{A}$ just transitions to some state of $x$ (since there may exist instantaneous transitions, at the instant, $\mathcal{A}$ may be in some state outside of $x$).

## 3.2 The notions of state-based opacity

In order to define state-based opacity, we need to specify a special subset $Q_S \subset Q$ of *secret states*. The notions of state-based opacity describe the ability of an RTA $\mathcal{A}$ forbidding an external intruder from making sure whether some secret state has been visited when the intruder observes timed output sequences generated by $\mathcal{A}$, given that the intruder knows full knowledge of the structure of $\mathcal{A}$. Before defining opacity formally, we specify a special class of states from which a secret state will definitely be reached through unobservable transitions. A state $q$ of RTA $\mathcal{A}$ is called *eventually secret* if either (1) $q$ is secret or (2) there is an unobservable path starting from $q$ and along each of such paths at least one secret state will be visited. Hence a state $q$ is not eventually secret if and only if (1) $q \notin Q_S$ and (2) either there is no unobservable path starting from $q$ or there is an unobservable path starting at $q$ without any secret state that either ends at a dead state or contains repetitive states.

▶ **Definition 4** (ISO). *An RTA $\mathcal{A}$ is called* initial-state opaque (with respect to $Q_S$) *if for every $\gamma \in \mathcal{L}(\mathcal{A})$, $\text{init}([\gamma]) \not\subset Q_S$.*

▶ **Definition 5** (CSO). *An RTA $\mathcal{A}$ is called* current-state opaque (with respect to $Q_S$) *if every $\gamma \in \mathcal{L}(\mathcal{A})$, in $\mathcal{M}(\mathcal{A}, \gamma)$ there exists at least one non-eventual-secret state of $\mathcal{A}$.*

▶ **Definition 6** (InfSO). *An RTA $\mathcal{A}$ is called* infinite-step opaque (with respect to $Q_S$) *if for all $\gamma_1\gamma_2 \in \mathcal{L}(\mathcal{A})$ such that $|\gamma_2| \geqslant 1$, $\text{interm}(\gamma_1, \gamma_2)$ contains a state $q$ such that there is a run $q \to q' \xrightarrow{e/t} q''$ with weight $\text{init}_R(\gamma_2) - \text{last}_R(\gamma_1)$, where $q \to q'$ is unobservable and contains no secret state, $e$ is observable and $\ell(e) = \text{init}_L(\gamma_2)$, $q'' \in \text{interm}(\gamma_1 \,\text{init}(\gamma_2), \gamma_2 \backslash \text{init}(\gamma_2))$.*

▶ **Definition 7** (KSO). *Let $K$ be in $\mathbb{Z}_+$. An RTA $\mathcal{A}$ is called* $K$-step opaque (with respect to $Q_S$) *if for all $\gamma_1\gamma_2 \in \mathcal{L}(\mathcal{A})$ such that $1 \leqslant |\gamma_2| \leqslant K$, $\text{interm}(\gamma_1, \gamma_2)$ contains a state $q$ such that there is a run $q \to q' \xrightarrow{e/t} q''$ with weight $\text{init}_R(\gamma_2) - \text{last}_R(\gamma_1)$, where $q \to q'$ is unobservable and contains no secret state, $e$ is observable and $\ell(e) = \text{init}_L(\gamma_2)$, $q'' \in \text{interm}(\gamma_1 \,\text{init}(\gamma_2), \gamma_2 \backslash \text{init}(\gamma_2))$.*

Intuitively, when an intruder observes a timed output sequence generated by an RTA $\mathcal{A}$, if $\mathcal{A}$ is initial-state opaque, then the intruder cannot make sure whether the initial state is secret; is current-state opaque, then the intruder cannot make sure after the last observable event occurred (observing $\text{last}_L(\gamma)$) and before any new observable event occurs, whether a

secret state has been visited (note that the "current time" here means the weight of any run of $[\gamma]$, so is no less than $\text{last}_R(\gamma)$ and $>$ may hold); infinite-step opaque, then the intruder cannot make sure whether a secret state was visited after observing $\text{last}_L(\gamma_1)$ and before observing $\text{init}_L(\gamma_2)$; and $K$-step opaque, then the intruder cannot make sure whether the state prior to at most $K$ observed outputs is secret. Different notions have different privacy levels, so they may have different applications.

## 3.3 The notion of observer

In this subsection we define and compute a notion of *observer* $\mathcal{A}_{\text{obs}}$ to concatenate current-state estimates along timed output sequences generated by $\mathcal{A}$ in 2-EXPTIME in the size of $\mathcal{A}$. Later, we will use $\mathcal{A}_{\text{obs}}$ to give a necessary and sufficient condition for CSO. Before defining $\mathcal{A}_{\text{obs}}$, we need to define a notion of *pre-observer* $\mathcal{A}^{\text{pre}}_{\text{obs}}$.

▶ **Definition 8.** *For an RTA $\mathcal{A}$, we define its* pre-observer *as a deterministic automaton*

$$\mathcal{A}^{\text{pre}}_{\text{obs}} = (X, \Sigma \times \mathbb{R}_{\geqslant 0}, x_0, \delta^{\text{pre}}_{\text{obs}}), \tag{6}$$

*where $X \subset 2^Q \backslash \{\varnothing\}$ is the state set, $\Sigma \times \mathbb{R}_{\geqslant 0}$ the alphabet, $x_0 = \mathcal{M}(\mathcal{A}, \epsilon) \in X$ the unique initial state, $\delta^{\text{pre}}_{\text{obs}} \subset X \times (\Sigma \times \mathbb{R}_{\geqslant 0}) \times X$ the transition relation. For all nonempty $x \subset Q$ different from $x_0$, $x \in X$ if and only if there is $\gamma \in (\Sigma \times \mathbb{R}_{\geqslant 0})^+$ such that $x = \mathcal{M}(\mathcal{A}, \gamma)$. For all $x, x' \in X$ and $(\sigma, t) \in \Sigma \times \mathbb{R}_{\geqslant 0}$, $(x, (\sigma, t), x') \in \delta^{\text{pre}}_{\text{obs}}$ if and only if $x' = \mathcal{M}(\mathcal{A}, (\sigma, t)|x)$.*

In Definition 8, after $\delta^{\text{pre}}_{\text{obs}}$ is recursively extended to $\delta^{\text{pre}}_{\text{obs}} \subset X \times (\Sigma \times \mathbb{R}_{\geqslant 0})^* \times X$, one has for all $x \in X$ and $(\sigma_1, t_1) \dots (\sigma_n, t_n) =: \gamma \in (\Sigma \times \mathbb{R}_{\geqslant 0})^+$, $(x_0, \gamma, x) \in \delta^{\text{pre}}_{\text{obs}}$ if and only if $\mathcal{M}(\mathcal{A}, \tau(\gamma)) = x$, where $\tau(\gamma)$ is defined in (3), i.e., $x$ is the set of states that $\mathcal{A}$ can be in when timed output sequence $\tau(\gamma)$ has just been generated.

Note that the alphabet $\Sigma \times \mathbb{R}_{\geqslant 0}$ is not finite, so we cannot compute the whole $\mathcal{A}^{\text{pre}}_{\text{obs}}$. Next, we define observer $\mathcal{A}_{\text{obs}}$ as a computable sub-automaton of $\mathcal{A}^{\text{pre}}_{\text{obs}}$.

▶ **Definition 9.** *For an RTA $\mathcal{A}$, consider its pre-observer* (8)*, we define its* observer *as a finite automaton*

$$\mathcal{A}_{\text{obs}} = (X, \Sigma_{\text{obs}}, x_0, \delta_{\text{obs}}), \tag{7}$$

*where $\Sigma_{\text{obs}}$ (resp., $\delta_{\text{obs}}$) is a finite subset of $\Sigma \times \mathbb{Q}_{\geqslant 0}$ (resp., $\delta^{\text{pre}}_{\text{obs}}$), such that if there exists a transition from $x \in X$ to $x' \in X$ in $\delta^{\text{pre}}_{\text{obs}}$ then at least one such transition belongs to $\delta_{\text{obs}}$.*

Note that for an RTA $\mathcal{A}$, its observer may not be unique, because $\Sigma_{\text{obs}}$ may not be unique; however, $X$ and $x_0$ must be unique. In Definition 9, after $\delta_{\text{obs}}$ is recursively extended to $\delta_{\text{obs}} \subset X \times (\Sigma_{\text{obs}})^* \times X$, one has for all $x \in X$ and $(\sigma_1, t_1) \dots (\sigma_n, t_n) =: \gamma \in (\Sigma_{\text{obs}})^+$, $(x_0, \gamma, x) \in \delta_{\text{obs}}$ if and only if $\mathcal{M}(\mathcal{A}, \tau(\gamma)) = x$.

▶ **Theorem 10.** *For an RTA $\mathcal{A}$, its observer $\mathcal{A}_{\text{obs}}$ can be computed in* 2-EXPTIME *in the size of $\mathcal{A}$.*

Here we only give a sketch of the proof, the entire proof is put in Appendix. The initial state $x_0 = \mathcal{M}(\mathcal{A}, \epsilon)$ is trivially computable in polynomial time. We then start from $x_0$, find all reachable states step by step together with the corresponding transitions, which is equivalent to checking for all $x_1, x_2 \subset Q$ and $\sigma \in \Sigma$, whether there is a transition $(x_1, (\sigma, t), x_2)$ for some $t \in \mathbb{Q}_{\geqslant 0}$. In addition, we require that for all $x_1, x_2, x_3 \subset Q$, if we find two transitions $(x_1, (\sigma, t), x_2)$ and $(x_1, (\sigma, t'), x_3)$ for some $t, t' \in \mathbb{Q}_{\geqslant 0}$, then $x_2 \subset x_3$ implies $x_3 \not\subset \mathcal{M}(\mathcal{A}, (\sigma, t)|x_1)$. This guarantees that if there exists a transition from $x_1 \subset Q$ to $x_2 \subset Q$ in $\mathcal{A}^{\text{pre}}_{\text{obs}}$, then there also exists a transition from $x_1 \subset Q$ to $x_2 \subset Q$ in $\mathcal{A}_{\text{obs}}$.

## 3.4 The notion of reverse observer

In this subsection we define and compute a notion of *reverse observer* ${}_{\mathsf{R}}\mathcal{A}_{\text{obs}}$ that will be used to verify ISO. To this end, we also need to define a notion of *pre-reverse observer* ${}_{\mathsf{R}}\mathcal{A}^{\text{pre}}_{\text{obs}}$. Similarly to observer $\mathcal{A}_{\text{obs}}$, ${}_{\mathsf{R}}\mathcal{A}_{\text{obs}}$ can also be computed in 2-EXPTIME in the size of $\mathcal{A}$.

For an RTA $\mathcal{A}$, a subset $x \subset Q$ of states, and a sequence $\gamma \in (\Sigma \times \mathbb{R}_{\geqslant 0})^+$, we define the *reverse-current-state estimate* as

$$\begin{aligned}\mathcal{M}^{\mathsf{R}}(\mathcal{A}, \gamma|x) := \{q \in Q | (\exists q' \in x)(\exists n \in \mathbb{Z}_+)(\exists m \in \mathbb{N}) \\ \left(\exists \text{ a run } \pi = q \xrightarrow{e_1/t_1} \cdots \xrightarrow{e_n/t_n} q_n \xrightarrow{e_{n+1}/0} \cdots \xrightarrow{e_{n+m}/0} q'\right) \\ [(e_n \in E_o) \wedge (e_{n+1} \dots e_{n+m} \in (E_{uo})^*) \wedge \ell(\tau(\pi)) = \gamma]\}. \end{aligned} \tag{8}$$

$\mathcal{M}^{\mathsf{R}}(\mathcal{A}, \gamma|x)$ denotes the subset of states of $Q$ starting from which at instant 0, $\mathcal{A}$ can generate timed output sequence $\gamma \in (\Sigma \times \mathbb{R}_{\geqslant 0})^+$ and can only be in any one state of $x$ at instant $\text{last}_R(\gamma)$.

▶ **Definition 11.** *For an RTA $\mathcal{A}$, we define its* pre-reverse observer *as a deterministic automaton*

$${}_{\mathsf{R}}\mathcal{A}^{\text{pre}}_{\text{obs}} = (X_{\mathsf{R}}, \Sigma \times \mathbb{R}_{\geqslant 0}, Q, {}_{\mathsf{R}}\delta^{\text{pre}}_{\text{obs}}), \tag{9}$$

*where $X_{\mathsf{R}} \subset 2^Q \backslash \{\varnothing\}$ is the state set, $\Sigma \times \mathbb{R}_{\geqslant 0}$ the alphabet, $Q$ the unique initial state, ${}_{\mathsf{R}}\delta^{\text{pre}}_{\text{obs}} \subset X_{\mathsf{R}} \times (\Sigma \times \mathbb{R}_{\geqslant 0}) \times X_{\mathsf{R}}$ the transition relation. For all $x, x' \in X_{\mathsf{R}}$ and $(\sigma, t) \in \Sigma \times \mathbb{R}_{\geqslant 0}$, $(x, (\sigma, t), x') \in {}_{\mathsf{R}}\delta^{\text{pre}}_{\text{obs}}$ if and only if $x' = \mathcal{M}^{\mathsf{R}}(\mathcal{A}, (\sigma, t)|x)$, i.e., $x'$ is the subset of states of $Q$ starting from which at instant 0, $\mathcal{A}$ can generate timed output sequence $(\sigma, t)$ and can only be in any one state of $x$ at instant $t$.*

▶ **Definition 12.** *For an RTA $\mathcal{A}$, consider its pre-reverse observer* (9)*, we define its* reverse observer *as a finite automaton*

$${}_{\mathsf{R}}\mathcal{A}_{\text{obs}} = (X_{\mathsf{R}}, {}_{\mathsf{R}}\Sigma_{\text{obs}}, Q, {}_{\mathsf{R}}\delta_{\text{obs}}), \tag{10}$$

*where ${}_{\mathsf{R}}\Sigma_{\text{obs}}$ (resp., ${}_{\mathsf{R}}\delta_{\text{obs}}$) is a finite subset of $\Sigma \times \mathbb{Q}_{\geqslant 0}$ (resp., ${}_{\mathsf{R}}\delta^{\text{pre}}_{\text{obs}}$) such that for all $x_1, x_2 \in X_{\mathsf{R}}$, if there is a transition from $x_1$ to $x_2$ in ${}_{\mathsf{R}}\delta^{\text{pre}}_{\text{obs}}$ then at least one such transition belongs to ${}_{\mathsf{R}}\delta_{\text{obs}}$.*

In ${}_{\mathsf{R}}\mathcal{A}_{\text{obs}}$, ${}_{\mathsf{R}}\Sigma_{\text{obs}}$ and ${}_{\mathsf{R}}\delta_{\text{obs}}$ may not be unique but must be finite. The following result follows from a similar proof compared with that of Theorem 10.

▶ **Theorem 13.** *For an RTA $\mathcal{A}$, its reverse observer ${}_{\mathsf{R}}\mathcal{A}_{\text{obs}}$ can be computed in* 2-EXPTIME *in the size of $\mathcal{A}$.*

## 3.5 Necessary and sufficient conditions for notions of state-based opacity

In this subsection, we use the notions of observer and reverse observer to give necessary and sufficient conditions for the four notions of opacity.

▶ **Theorem 14.** *Consider an RTA $\mathcal{A}$. $\mathcal{A}$ is initial-state opaque if and only if in reverse observer ${}_{\mathsf{R}}\mathcal{A}_{\text{obs}}$, for every reachable state $x$, if $x \cap Q_0 \neq \varnothing$ then $x \cap Q_0 \nsubseteq Q_S$; $\mathcal{A}$ is current-state opaque if and only if in observer $\mathcal{A}_{\text{obs}}$, every reachable state $x$ contains at least one non-eventual-secret state of $\mathcal{A}$.*

**Proof.** Consider an arbitrary $\gamma \in \mathcal{L}(\mathcal{A})$.

By definition, $\mathrm{init}([\gamma]) = \mathcal{M}^{\mathsf{R}}(\mathcal{A}, \gamma|Q) \cap Q_0$; and in reverse observer ${}_{\mathsf{R}}\mathcal{A}_{\mathrm{obs}}$, there exists $\gamma' \in ({}_{\mathsf{R}}\Sigma_{\mathrm{obs}})^*$ such that $(Q, \gamma', \mathcal{M}^{\mathsf{R}}(\mathcal{A}, \gamma|Q)) \in {}_{\mathsf{R}}\delta_{\mathrm{obs}}$. For every $\gamma'' \in ({}_{\mathsf{R}}\Sigma_{\mathrm{obs}})^*$ and state $x$ of ${}_{\mathsf{R}}\mathcal{A}_{\mathrm{obs}}$ such that $(Q, \gamma'', x) \in {}_{\mathsf{R}}\delta_{\mathrm{obs}}$, $x = \mathcal{M}^{\mathsf{R}}(\mathcal{A}, \tau((\gamma'')^{\mathsf{R}})|Q)$. Hence the set $\{x \cap Q_0 | x \text{ is reachable in } {}_{\mathsf{R}}\mathcal{A}_{\mathrm{obs}}\}$ is equal to the set $\{\mathrm{init}([\gamma]) | \gamma \in \mathcal{L}(\mathcal{A})\}$. Then, $\mathcal{A}$ is initial-state opaque if and only if for every reachable state $x$ of ${}_{\mathsf{R}}\mathcal{A}_{\mathrm{obs}}$, if $x \cap Q_0 \neq \varnothing$, then $x \cap Q_0 \not\subset Q_S$.

By definition, there exists $\gamma' \in (\Sigma_{\mathrm{obs}})^*$ such that $(x_0, \gamma', \mathcal{M}(\mathcal{A}, \gamma)) \in \delta_{\mathrm{obs}}$. Conversely, for every $\gamma'' \in (\Sigma_{\mathrm{obs}})^*$ and $x \in X$ such that $(x_0, \gamma'', x) \in \delta_{\mathrm{obs}}$, one has $x = \mathcal{M}(\mathcal{A}, \tau(\gamma''))$, where $\tau(\gamma'') \in \mathcal{L}(\mathcal{A})$. Then, $\mathcal{A}$ is current-state opaque if and only if every reachable state of $\mathcal{A}_{\mathrm{obs}}$ contains at least one non-eventual-secret state of $\mathcal{A}$. ◀

By Theorem 10, Theorem 13, and Theorem 14, the initial-state opacity and current-state opacity of an RTA $\mathcal{A}$ can be verified in time linear in the size of ${}_{\mathsf{R}}\mathcal{A}_{\mathrm{obs}}$ and $\mathcal{A}_{\mathrm{obs}}$, respectively, hence in 2-EXPTIME in the size of $\mathcal{A}$.

▶ **Theorem 15.** *Consider an RTA $\mathcal{A}$ and $K \in \mathbb{Z}_+$. $\mathcal{A}$ is infinite-step opaque if and only if for every reachable state $x$ of observer $\mathcal{A}_{\mathrm{obs}}$ and every transition $(x'', (\sigma, t), x')$ of reverse observer ${}_{\mathsf{R}}\mathcal{A}_{\mathrm{obs}}$ with $x''$ being reachable in ${}_{\mathsf{R}}\mathcal{A}_{\mathrm{obs}}$, if $x \cap x' \neq \varnothing$, then $x \cap x'$ contains a state $q \in Q$ such that there is a run $q \to q' \xrightarrow{e/t'} q''$ with weight $t$, where $q \to q'$ is unobservable and contains no secret state of $Q_S$, $e$ is observable and $\ell(e) = \sigma$, $q'' \in x''$; $\mathcal{A}$ is $K$-step opaque if and only if the above necessary and sufficient condition for InfSO holds, and $x''$ additionally satisfies that there is $\gamma \in ({}_{\mathsf{R}}\Sigma_{\mathrm{obs}})^*$ such that $|\gamma| \leqslant K - 1$ and $(Q, \gamma, x'') \in {}_{\mathsf{R}}\delta_{\mathrm{obs}}$.*

**Proof.** By definition of pre-reverse observer ${}_{\mathsf{R}}\mathcal{A}^{\mathrm{pre}}_{\mathrm{obs}}$, one sees that for all $\gamma_1\gamma_2 \in \mathcal{L}(\mathcal{A})$ such that $|\gamma_2| \geqslant 1$, $\mathrm{interm}(\gamma_1, \gamma_2) = \mathcal{M}(\mathcal{A}, \gamma_1) \cap x_1$, where $x_1$ satisfies $(Q, (\tau^{-1}(\gamma_2 - \mathrm{last}_R(\gamma_1)))^{\mathsf{R}}, x_1) \in {}_{\mathsf{R}}\delta^{\mathrm{pre}}_{\mathrm{obs}}$. Choose $x_2 \in X_{\mathsf{R}}$ such that $(x_2, \mathrm{init}(\gamma_2) - \mathrm{last}_R(\gamma_1), x_1) \in {}_{\mathsf{R}}\delta^{\mathrm{pre}}_{\mathrm{obs}}$. Then by definition of InfSO, one has $\mathcal{A}$ is infinite-step opaque if and only if for all such $\gamma_1\gamma_2$, $x_1$, and $x_2$, if $\mathcal{M}(\mathcal{A}, \gamma_1) \cap x_1 \neq \varnothing$, then $\mathcal{M}(\mathcal{A}, \gamma_1) \cap x_1$ contains a state $q \in Q$ such that there is a run $q \to q' \xrightarrow{e/t'} q''$ with weight $\mathrm{init}_R(\gamma_2) - \mathrm{last}_R(\gamma_1)$, where $q \to q'$ is unobservable and contains no secret state of $Q_S$, $e$ is observable and $\ell(e) = \mathrm{init}_L(\gamma_2)$, $q'' \in x_2$. By definitions of observer $\mathcal{A}_{\mathrm{obs}}$ and reverse observer ${}_{\mathsf{R}}\mathcal{A}_{\mathrm{obs}}$, there exist $\bar{\gamma}_1, \bar{\gamma}_2 \in (\Sigma \times \mathbb{Q}_{\geqslant 0})^*$ such that $|\bar{\gamma}_1| = |\gamma_1|$, $|\bar{\gamma}_2| = |\gamma_2|$, $(x_0, \bar{\gamma}_1, \mathcal{M}(\mathcal{A}, \gamma_1)) \in \delta_{\mathrm{obs}}$, $(Q, \bar{\gamma}_2, x_1) \in {}_{\mathsf{R}}\delta_{\mathrm{obs}}$, and $(x_2, \mathrm{last}(\bar{\gamma}_2), x_1) \in {}_{\mathsf{R}}\delta_{\mathrm{obs}}$. Then one has the necessary and sufficient condition for InfSO holds. Similarly one has the necessary and sufficient condition for KSO also holds. ◀

We next give an upper bound for $K$.

▶ **Proposition 16.** *Consider an RTA $\mathcal{A}$ and a positive integer $K \in \mathbb{Z}_+$. $\mathcal{A}$ is $K$-step opaque if and only if it is $\min\{K, 2^{|Q|}\}$-step opaque.*

**Proof.** By definition, if $\mathcal{A}$ is $K$-step opaque, then it is $K'$-step opaque for all $1 \leqslant K' < K$. Then the "only if" part holds.

Next we prove the "if" part. Assume $K > 2^{|Q|}$ and $\mathcal{A}$ is not $K$-step opaque. By Theorem 15, for observer $\mathcal{A}_{\mathrm{obs}}$ and reverse observer ${}_{\mathsf{R}}\mathcal{A}_{\mathrm{obs}}$, there exist $\gamma_1 \in (\Sigma_{\mathrm{obs}})^*$, $\gamma_2 \in ({}_{\mathsf{R}}\Sigma_{\mathrm{obs}})^*$, $x_1 \in X$, $x_2, x_2' \in X_{\mathsf{R}}$, such that $(x_0, \gamma_1, x_1) \in \delta_{\mathrm{obs}}$, $(Q, \gamma_2, x_2) \in {}_{\mathsf{R}}\delta_{\mathrm{obs}}$, $(x_2', \mathrm{last}(\gamma_2), x_2) \in {}_{\mathsf{R}}\delta_{\mathrm{obs}}$, and $1 \leqslant |\gamma_2| \leqslant K$; $x_1 \cap x_2$ is nonempty and does not contain a state $q$ of $Q$ such that there is a run $q \to q' \xrightarrow{e/t'} q''$ with weight $\mathrm{last}_R(\gamma_2)$, where $q \to q'$ is unobservable and contains no secret state of $Q_S$, $e$ is observable and $\ell(e) = \mathrm{last}_L(\gamma_2)$, $q'' \in x_2'$. Because ${}_{\mathsf{R}}\mathcal{A}_{\mathrm{obs}}$ has at most $2^{|Q|}$ states, there exists $\gamma_3 \in ({}_{\mathsf{R}}\Sigma_{\mathrm{obs}})^*$ such that $|\gamma_3| \leqslant 2^{|Q|} - 1$ and $(Q, \gamma_3, x_2') \in {}_{\mathsf{R}}\delta_{\mathrm{obs}}$. Then also by Theorem 15, $\mathcal{A}$ is not $2^{|Q|}$-step opaque. ◀

By Theorem 10, Theorem 13, Theorem 15, and Proposition 16, for any $K \in \mathbb{Z}_+$, the infinite-step opacity and $K$-step opacity of an RTA $\mathcal{A}$ can be verified in 2-EXPTIME in the size of $\mathcal{A}$.

▶ **Example 17.** Consider the toy RTA $\mathcal{A}_1$ in Figure 1. When $\mathcal{A}_1$ starts at $q_0$, and $a$ occurs at instant between 1 and 2, $\mathcal{A}_1$ can transition to either $q_1$ or $q_2$; but if $a$ occurs at instant in $(2, 3]$, $\mathcal{A}_1$ can only transition to $q_1$.

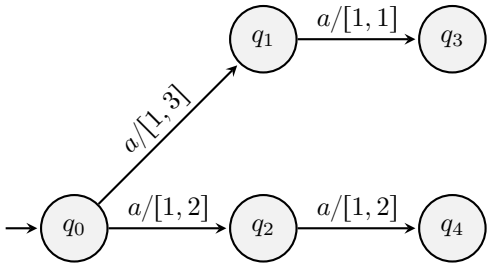



**Figure 1** An RTA $\mathcal{A}_1$, where a state with an arrow from nowhere denotes an initial state, e.g., $q_0$; $a$ is an observable event, $\ell(a) = a$.

One of its observers is shown in Figure 2.

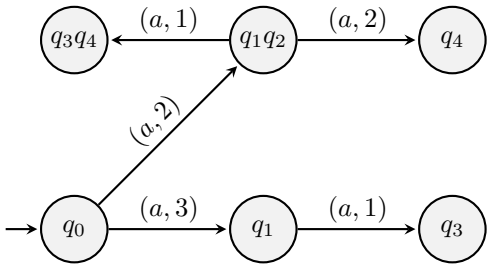



**Figure 2** One observer $\mathcal{A}_{1\,\mathrm{obs}}$ of RTA $\mathcal{A}_1$ in Figure 1.

One of its reverse observers is shown in Figure 3.

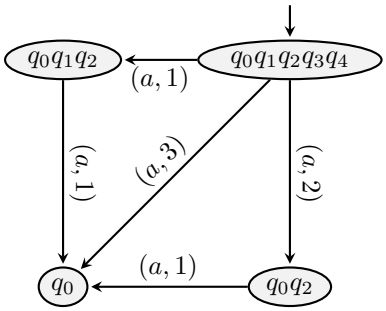



**Figure 3** One reverse observer ${}_{\mathrm{R}}\mathcal{A}_{1\,\mathrm{obs}}$ of RTA $\mathcal{A}_1$ in Figure 1.

One sees two runs $q_0 \xrightarrow{a/2} q_1 \xrightarrow{a/1} q_3$ and $q_0 \xrightarrow{a/1} q_2 \xrightarrow{a/1} q_4$ of $\mathcal{A}_1$, where $2 \in \mu(a)_{q_0q_1} = [1, 3]$.

Now assume $q_3$ is secret, all the other states are non-secret. By definition, only $q_3$ is eventually secret, because there is no unobservable path starting from any other state. By observer $\mathcal{A}_{1\,\mathrm{obs}}$ and Theorem 14, $\mathcal{A}_1$ is not current-state opaque with respect to $\{q_3\}$ because

there is a reachable state $\{q_3\}$ in $\mathcal{A}_{1\,\mathrm{obs}}$ that only contains eventually secret states of $\mathcal{A}_1$ (that is, $q_3$). On the other hand, after observing $(a,3)(a,4)$, one can make sure that $\mathcal{A}_1$ is in state $q_3$ by $\mathcal{A}_{1\,\mathrm{obs}}$.

Now assume only $q_1$ is secret. In observer $\mathcal{A}_{1\,\mathrm{obs}}$ there is a reachable state $\{q_1\}$, in reverse observer ${}_{\mathsf{R}}\mathcal{A}_{1\,\mathrm{obs}}$ there is a reachable transition $\{q_0,\ldots,q_4\} \xrightarrow{(a,1)} \{q_0,q_1,q_2\}$. One has $\{q_1\} \cap \{q_0,q_1,q_2\} = \{q_1\}$, which contains only secret states. Then by Theorem 15, $\mathcal{A}_1$ is not infinite-step opaque with respect to $\{q_1\}$.

## 4 Conclusion

In this paper, we formulated four notions of state-based opacity for real-time automata, and proved that the four notions are decidable in 2-EXPTIME by defining notions of observer and reverse observer and computing them in 2-EXPTIME. The lower bounds for verifying the four notions are not known.

In addition, one can see from Theorem 10 and Theorem 13 that if an RTA $\mathcal{A}$ has no unobservable cycle, then its observers and reverse observers can be computed in EXPTIME in the size of $\mathcal{A}$ without using the ERL problem. Hence by Theorem 14 and Theorem 15, the four notions of opacity can also be verified in EXPTIME in this special case.

## References


1 J. Balun and T. Masopust. Comparing the notions of opacity for discete-event systems, 2021. URL: https://arxiv.org/abs/2102.02889.

2 B. Bérard, S. Haar, S. Schmitz, and S. Schwoon. The complexity of diagnosability and opacity verification for Petri nets. *Fundamenta Informaticae*, 161(4):317–349, 2018.

3 J. W. Bryans, M. Koutny, L. Mazaré, and P. Y. A. Ryan. Opacity generalised to transition systems. *International Journal of Information Security*, 7(6):421–435, November 2008. doi:10.1007/s10207-008-0058-x.

4 F. Cassez. The dark side of timed opacity. In Jong Hyuk Park, Hsiao-Hwa Chen, Mohammed Atiquzzaman, Changhoon Lee, Tai-hoon Kim, and Sang-Soo Yeo, editors, *Advances in Information Security and Assurance*, pages 21–30, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

5 F. Cassez, J. Dubreil, and H. Marchand. Dynamic observers for the synthesis of opaque systems. In *Automated Technology for Verification and Analysis*, pages 352–367, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

6 S. Chédor, C. Morvan, S. Pinchinat, and H. Marchand. Diagnosis and opacity problems for infinite state systems modeled by recursive tile systems. *Discrete Event Dynamic Systems*, 25(1-2):271–294, 2014.

7 F. Lin. Opacity of discrete event systems and its applications. *Automatica*, 47(3):496–503, 2011. doi:10.1016/j.automatica.2011.01.002.

8 L. Mazaré. Using unification for opacity properties. In *Proceedings of the Workshop on Issues in the Theory of Security (WITS'04)*, pages 165–176, 2004.

9 M. Nykänen and E. Ukkonen. The exact path length problem. *Journal of Algorithms*, 42(1):41–53, 2002. doi:10.1006/jagm.2001.1201.

10 A. Saboori and C. N. Hadjicostis. Notions of security and opacity in discrete event systems. In *2007 46th IEEE Conference on Decision and Control*, pages 5056–5061, December 2007. doi:10.1109/CDC.2007.4434515.

11 A. Saboori and C. N. Hadjicostis. Verification of $K$-step opacity and analysis of its complexity. In *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, pages 205–210, 2009. doi:10.1109/CDC.2009.5400083.

12 A. Saboori and C. N. Hadjicostis. Verification of infinite-step opacity and complexity considerations. *IEEE Transactions on Automatic Control*, 57(5):1265–1269, May 2012. doi:10.1109/TAC.2011.2173774.

13 A. Saboori and C. N. Hadjicostis. Verification of initial-state opacity in security applications of discrete event systems. *Information Sciences*, 246:115–132, 2013. doi:10.1016/j.ins.2013.05.033.

14 A. Schrijver. *Theory of Linear and Integer Programming*. John Wiley & Sons, Inc., USA, 1986.

15 M. Sipser. *Introduction to the Theory of Computation*. International Thomson Publishing, 1st edition, 1996.

16 Y. Tong, Z. Li, C. Seatzu, and A. Giua. Decidability of opacity verification problems in labeled Petri net systems. *Automatica*, 80:48–53, 2017. doi:10.1016/j.automatica.2017.01.013.

17 L. Wang and N. Zhan. *Decidability of the Initial-State Opacity of Real-Time Automata*, pages 44–60. Springer International Publishing, Cham, 2018. doi:10.1007/978-3-030-01461-2_3.

18 L. Wang, N. Zhan, and J. An. The opacity of real-time automata. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 37(11):2845–2856, 2018. doi:10.1109/TCAD.2018.2857363.

19 Y. Wu and S. Lafortune. Comparative analysis of related notions of opacity in centralized and coordinated architectures. *Discrete Event Dynamic Systems*, 23(3):307–339, September 2013. doi:10.1007/s10626-012-0145-z.

20 X. Yin and S. Lafortune. A new approach for the verification of infinite-step and $K$-step opacity using two-way observers. *Automatica*, 80:162–171, 2017. doi:10.1016/j.automatica.2017.02.037.


# A Appendix

**Proof.** (of Theorem 10) The initial state $x_0 = \mathcal{M}(\mathcal{A}, \epsilon)$ is trivially computable in polynomial time. We then start from $x_0$, find all reachable states step by step together with the corresponding transitions, which is equivalent to checking for all $x_1, x_2 \subset Q$ and $\sigma \in \Sigma$, whether there is a transition $(x_1, (\sigma, t), x_2)$ for some $t \in \mathbb{Q}_{\geqslant 0}$. If it does exist, then $(x_1, (\sigma, t), x_2)$ is a transition of $\mathcal{A}_{\text{obs}}$, otherwise there is no transition $(x_1, (\sigma, t'), x_2)$ for any $t' \in \mathbb{Q}_{\geqslant 0}$ in $\mathcal{A}_{\text{obs}}$, and furthermore there is no transition $(x_1, (\sigma, t''), x_2)$ for any $t'' \in \mathbb{R}_{\geqslant 0}$ in $\mathcal{A}_{\text{obs}}^{\text{pre}}$, because $\mathbb{Q}_{\geqslant 0}$ is dense in $\mathbb{R}_{\geqslant 0}$. In addition, we require that for all $x_1, x_2, x_3 \subset Q$, if we find two transitions $(x_1, (\sigma, t), x_2)$ and $(x_1, (\sigma, t'), x_3)$ for some $t, t' \in \mathbb{Q}_{\geqslant 0}$, then $x_2 \subset x_3$ implies $x_3 \not\subset \mathcal{M}(\mathcal{A}, (\sigma, t)|x_1)$. This guarantees that if there exists a transition from $x_1 \subset Q$ to $x_2 \subset Q$ in $\mathcal{A}_{\text{obs}}^{\text{pre}}$, then there also exists a transition from $x_1 \subset Q$ to $x_2 \subset Q$ in $\mathcal{A}_{\text{obs}}$. The procedure for doing the above check is as follows.

Choose a state $x_1 = \{q_1, \ldots, q_n\} \in X$ that we have just computed, where $n \in \mathbb{Z}_+$, and $|x| = n$. Choose $\sigma \in \Sigma$. For each $i \in [\![1, n]\!]$, compute subautomaton $\mathcal{A}_{q_i}$ of $\mathcal{A}$ that consists of all paths of the form

$$q_i \xrightarrow{s_i^1} q_i^1 \xrightarrow{e_i} q_i^2 \tag{11}$$

such that $s_i^1 \in (E_{uo})^*$, $e_i \in E_o$, and $\ell(e_i) = \sigma$. Denote the set of all such $q_i^2$ by $\bar{x}_2$.

We next check for each $\varnothing \neq \tilde{x}_2 \subset \bar{x}_2$, whether $(x_1, (\sigma, t), \mathcal{M}(\mathcal{A}, \epsilon|\tilde{x}_2)) \in \delta_{\text{obs}}$ for some $t \in \mathbb{Q}_{\geqslant 0}$, in the order $|\tilde{x}_2|$ decreases. For every $\tilde{x}_2 \subsetneq \hat{x}_2 \subset \bar{x}_2$, if we have found a transition $(x_1, (\sigma, t'), \mathcal{M}(\mathcal{A}, \epsilon|\hat{x}_2))$ before checking $\tilde{x}_2$, then we must choose $t$ such that $\mathcal{M}(\mathcal{A}, \epsilon|\hat{x}_2) \not\subset \mathcal{M}(\mathcal{A}, (\sigma, t)|x_1)$. In order to finish the construction of $\mathcal{A}_{\text{obs}}$, we need to do the check for at most $2^{|Q|}2^{|Q|}$ times.

**[A]** For each $i \in [\![1,n]\!]$, denote the number of states $q_i^2$ shown in (11) by $i_2 \in \mathbb{N}$, and denote these states by $q_{i,1}^2, \dots, q_{i,i_2}^2$. Here one may have $i_2 = 0$, which implies that there is no path of the form (11) starting from $q_i$.

**[B]** Nondeterministically compute asynchronous product

$$\bigotimes_{i=1}^{1'_2} \mathcal{A}_{q_1} \otimes \cdots \otimes \bigotimes_{i=1}^{n'_2} \mathcal{A}_{q_n}, \tag{12}$$

where $i'_2 \leqslant i_2$, $i \in [\![1,n]\!]$, states $q_{1,1}^2, \dots, q_{1,1'_2}^2, \dots, q_{n,1}^2, \dots, q_{n,n'_2}^2$ are pairwise different and

$$\{q_{1,1}^2, \dots, q_{1,1'_2}^2, \dots, q_{n,1}^2, \dots, q_{n,n'_2}^2\} = \tilde{x}_2,$$

this also guarantees that $\sum_{i=1}^n i'_2 \leqslant |Q|$; the states of the product are

$$(q_{1,1}, \dots, q_{1,1'_2}, \dots, q_{n,1}, \dots, q_{n,n'_2}),$$

where $q_{i,1}, \dots, q_{i,i'_2}$ are states of $\mathcal{A}_{q_i}$, $i \in [\![1,n]\!]$; there is a transition

$$\begin{matrix}(q_{1,1}, \dots, q_{1,1'_2}, \dots, q_{n,1}, \dots, q_{n,n'_2}) \\ (q'_{1,1}, \dots, q'_{1,1'_2}, \dots, q'_{n,1}, \dots, q'_{n,n'_2})\end{matrix} \xrightarrow{(e_{1,1},\dots,e_{1,1'_2},\dots,e_{n,1},\dots,e_{n,n'_2})}$$

in product (12) if and only if either one of the two conditions holds.

**[a]** For some $i \in [\![1,n]\!]$ and $j \in [\![1,i'_2]\!]$, $q_{i,j} \xrightarrow{e_{i,j}} q'_{i,j}$ is an unobservable transition of $\mathcal{A}_{q_i}$, for all other pairs $(k,l)$, $e_{k,l}$ are qual to $\epsilon$, and $q_{k,l} = q'_{k,l}$. In this case, $\mu$ assigns to the transition a vector, where the $(i,j)$-component is the interval $\mu(e_{i,j})_{q_{i,j}q'_{i,j}}$, for all other $(k,l)$-components, $\mu(e_{k,l})_{q_{k,l}q'_{k,l}} = [0,0]$.

**[b]** For all $i \in [\![1,n]\!]$ and $j \in [\![1,i'_2]\!]$, $q_{i,j} \xrightarrow{e_{i,j}} q'_{i,j}$ is an observable transition of $\mathcal{A}_{q_i}$. In this case, $\mu$ assigns to the transition a vector, whether the $(i,j)$-components are intervals $\mu(e_{i,j})_{q_{i,j}q'_{i,j}}$.

**[C]** In product (12), guess transition

$$\begin{matrix}(q_{1,1}^1, \dots, q_{1,1'_2}^1, \dots, q_{n,1}^1, \dots, q_{n,n'_2}^1) \\ (q_{1,1}^2, \dots, q_{1,1'_2}^2, \dots, q_{n,1}^2, \dots, q_{n,n'_2}^2),\end{matrix} \xrightarrow{(\bar{e}_{1,1},\dots,\bar{e}_{1,1'_2},\dots,\bar{e}_{n,1},\dots,\bar{e}_{n,n'_2})}$$

where $\bar{e}_{1,1}, \dots, \bar{e}_{1,1'_2}, \dots, \bar{e}_{n,1}, \dots, \bar{e}_{n,n'_2}$ are observable (i.e., item (Bb) is satisfied). Then check in product (12), whether there exists a run $\pi_1$ in

$$\begin{matrix}(q_{1,1}^1, \dots, q_{1,1'_2}^1, \dots, q_{n,1}^1, \dots, q_{n,n'_2}^1) \\ (q_{1,1}^2, \dots, q_{1,1'_2}^2, \dots, q_{n,1}^2, \dots, q_{n,n'_2}^2)\end{matrix} \xrightsquigarrow{(\bar{e}_{1,1},\dots,\bar{e}_{1,1'_2},\dots,\bar{e}_{n,1},\dots,\bar{e}_{n,n'_2})}$$

and an unobservable run $\pi_2$ in

$$(\underbrace{q_1, \dots, q_1}_{1'_2}, \dots, \underbrace{q_n, \dots, q_n}_{n'_2}) \rightsquigarrow (q_{1,1}^1, \dots, q_{1,1'_2}^1, \dots, q_{n,1}^1, \dots, q_{n,n'_2}^1) \tag{13}$$

such that the weight of $\pi_2\pi_1$ has equal components that are equal to a rational number, which actually corresponds to a positive instance $(\sum_{i=1}^n i'_2, (12), (\underbrace{q_1, \dots, q_1}_{1'_2}, \dots, \underbrace{q_n, \dots, q_n}_{n'_2}),$

$(q^1_{1,1}, \ldots, q^1_{1,1'_2}, \ldots, q^1_{n,1}, \ldots, q^1_{n,n'_2}), (q^2_{1,1}, \ldots, q^2_{1,1'_2}, \ldots, q^2_{n,1}, \ldots, q^2_{n,n'_2}))$ of the CLER problem (Problem 2). If Yes, then the weight is denoted by $t \in \mathbb{Q}_{\geqslant 0}$, and we find a transition

$$(x_1, (\sigma, t), \mathcal{M}(\mathcal{A}, \epsilon|\tilde{x}_2)) \tag{14}$$

of $\mathcal{A}_{\text{obs}}$.

We need to do the above check (C) for at most $2^{|Q|}2^{|Q|}|Q|^{|Q|} = 2^{2|Q|^2 \log |Q|}$ times (corresponding to nondeterministic computations of product (12)). Each check can be done by solving the corresponding CLER problem (Problem 2), and hence can be done in NP in the size $O((|Q|^{|Q|})^2(|E_o|^{|E_o|} + |Q||E_{uo}|)) = O(2^{|Q|^2 \log |Q|}(2^{|E_o| \log |E_o|} + |Q||E_{uo}|))$ of the product (12) by Lemma 3. Hence, the total time consumption of computing $\mathcal{A}_{\text{obs}}$ is 2-EXPTIME in the size of $\mathcal{A}$. ◀