A Uniquely Ergodic Cellular Automaton[☆]

Ilkka Törmä[a]

[a] *TUCS – Turku Centre for Computer Science*
Department of Mathematics and Statistics
20014 University of Turku, Finland
+358 2 333 6012

Abstract

We construct a one-dimensional uniquely ergodic cellular automaton which is not nilpotent. This automaton can perform asymptotically infinitely sparse computation, which nevertheless never disappears completely. The construction builds on the self-simulating automaton of Gács. We also prove related results of dynamical and computational nature, including the undecidability of unique ergodicity, and the undecidability of nilpotency in uniquely ergodic cellular automata.

Keywords: cellular automata, nilpotency, dynamical system, ergodic theory, unique ergodicity

1. Introduction

Cellular automata form a class of discrete dynamical systems that are simple to define but often surprisingly hard to analyze. They consist of an infinite array of cells, arranged in a regular lattice of one or more dimensions, each of which contains a state drawn from a finite set. The dynamics is given by a local function that synchronously assigns each cell a new state based on the previous states of itself and its neighbors. Cellular automata can be used as idealized models of many physical and biological processes, or as massively parallel computing devices, and their history goes back to the 60's [12].

Since cellular automata are so discrete and finitely definable, there are many natural decidability problems concerning their dynamical properties. The most classical result in this area is the undecidability of nilpotency [13]. A cellular automaton is nilpotent if there is a number $n \in \mathbb{N}$ such that, regardless of the initial contents of the cells, every cell is in the same 'blank' state after n timesteps. There are several variants of this property, some of which are equivalent to nilpotency, and some of which are not. If n can depend on the initial configuration and on an individual cell, then the automaton is nilpotent (proved in [9] for one-dimensional automata, and generalized in [18] to all dimensions).

[☆]Research supported by the Academy of Finland Grant 131558

In [2], a notion called μ-quasi-nilpotency was defined for a probability measure μ, using the concept of μ-limit sets defined in [14]. Simple examples show that for many natural measures μ, there exist μ-quasi-nilpotent cellular automata which are not nilpotent.

In this article, we consider the notion of *unique ergodicity*, which originates from ergodic theory. A dynamical system is uniquely ergodic if there is a unique probability measure on the state space which is invariant under the dynamics. For cellular automata with a blank state, this is equivalent to the condition that every cell of every initial configuration asymptotically spends only a negligible number of timesteps in a non-blank state. Our main result is the construction of a uniquely ergodic cellular automaton which is not nilpotent. We also define the notion of *asymptotic nilpotency in density*, which means that the density of non-blank cells of any initial configuration converges to 0, as the cellular automaton is applied repeatedly. Our uniquely ergodic automaton also has this property, while the two notions are independent in general.

Our construction is based on the error-resistant cellular automaton of Gács [7] (the article is extremely long and complicated, see [8] for a more reader-friendly outline). In particular, we use the powerful and adaptable tool of *programmatic self-simulation*. Self-similarity itself is not a new method in the field of symbolic dynamics and cellular automata, since the aperiodic tileset of Robinson, together with his proof of undecidability of the tiling problem, is based on geometric self-similarity [17]. Another classical example is the tileset of Mozes, which realizes a two-dimensional substitution system [15]. Programmatic self-simulation differs from these in that the self-similarity does not arise from geometric constructs, but from the system explicitly computing its own local structure and simulating it on a larger scale. In recent years, programmatic self-simulation has been successfully applied to numerous problems, for example to realize every effective closed one-dimensional subshift by a tiling system, and to construct a tiling system whose every valid tiling has high Kolmogorov complexity (see [6] and references therein). The possible topological entropies of cellular automata in all dimensions have also been characterized using the method [11].

The main idea of this paper is the following. We construct a cellular automaton which realizes approximate programmatic self-simulation. This means that in the time evolution of certain well-formed configurations, the cells are organized into 'macro-cells' that collectively behave similarly to the original system, so the macro-cells are again organized into 'macro-macro-cells' and so on. We construct the automaton so that it is *sparse* in the following sense: any given cell will spend at least a certain percentage of its time in the blank state. Each simulation level is also sparser than the previous, so that the macro-cells of a very high simulation level are blank almost all the time. Those macro-cells that represent blank cells consist of blank cells themselves, so that in the limit, every cell is almost always blank. We also guarantee that all patterns except the correct simulation structure are destroyed and replaced by blank cells. The construction is split into two relatively independent parts: a computable transformation that turns any cellular automaton into a sparse version of itself, and

a general self-simulating automaton on which we apply the transformation. The main reason for the complexity of the construction is that programmatic self-simulation is inherently very complicated and its implementation contains lots of technical details, even more so when extra conditions are required from the system. The length of our proof and the amount of definitions, on the other hand, are mainly due to the last condition that all incorrect structure must eventually disappear.

This paper is organized as follows. Section 2 consists of the (mostly) standard definitions and notation needed in this article. Section 3 contains the characterization of unique ergodicity for cellular automata with a quiescent state, and is the only section containing general ergodic theory. A reader not interested in ergodic theory may skip the proofs of this section, and use our characterization as the definition of unique ergodicity for cellular automata. In Section 4, we define several notions related to simulations between cellular automata, and prove their basic properties. Since our construction uses programmatic self-simulation, and involves quite complicated automata, Section 5 presents a simple programming language for expressing CA rules. We also define CA transformations, which are functions operating on these programs. Section 6 describes a certain CA transformation that performs sparsification, that is, modifies its input automaton so that all cells spend only a small (but positive) fraction of their time in non-blank states.

Section 7 describes a family of cellular automata that perform universal programmatic simulation. This is not a formal notion, but refers to the fact that the simulation is carried out using a Turing machine that interprets the program of the target automaton. In Section 8, these universal simulators are used to realize amplification, which is a form of approximate self-simulation parameterized by a CA transformation. Essentially, amplification produces a cellular automaton that programmatically simulates a transformed version of itself. In Section 9, we apply amplification to the sparsification transformation, obtaining an 'infinitely sparse' cellular automaton, and prove its unique ergodicity, the main result of this article. Section 10 contains some further definitions and results that can be proved using our amplifier construction, including asymptotic nilpotency in density, the undecidability of unique ergodicity, and the undecidability of nilpotency in the class of uniquely ergodic cellular automata. Finally, Section 11 contains our conclusions and some future research directions.

This work is based on the author's Master's thesis [19].

2. Definitions

Let Σ be a finite set, called the *state set*. For a set $D \subset \mathbb{Z}$ (or $D \subset \mathbb{Z}^2$), a function $s : D \to \Sigma$ is called a *pattern over* Σ *with domain* D. A pattern whose domain is an interval (product of intervals) is called a *word* (a *rectangle*, respectively), and a *cell* is a pattern whose domain is a singleton. Translates of a pattern are sometimes deemed equal, and sometimes not; the choice should always be clear from the context. In particular, by abuse of language and notation, cells are sometimes treated as elements of Σ. A pattern with domain

$\mathbb{Z}$ ($\mathbb{Z}^2$) is called a *one-dimensional configuration* (*two-dimensional configuration*, respectively). The sets of one- and two-dimensional configurations are denoted $\Sigma^{\mathbb{Z}}$ and $\Sigma^{\mathbb{Z}^2}$, respectively. If $x \in \Sigma^{\mathbb{Z}}$ and $i \in \mathbb{Z}$, we denote by x_i the i'th cell of x, and if $D \subset \mathbb{Z}$, we denote by x_D the restriction of x to D. For a two-dimensional configuration $\eta \in \Sigma^{\mathbb{Z}^2}$ and $(i,t) \in \mathbb{Z}^2$, we denote by η_i^t the cell of η at (i,t), and if $D, E \subset \mathbb{Z}$, we denote by η_D^E the restriction of η to $D \times E \subset \mathbb{Z}^2$. Also, for $i, t \in \mathbb{Z}$, we denote by $\eta_i = \eta_i^{\mathbb{Z}}$ the i'th column of η, and by $\eta^t = \eta_{\mathbb{Z}}^t$ the t'th row of η. For a rectangular pattern $P \in \Sigma^{km \times \ell n}$ and a function $\pi : \Sigma^{k \times \ell} \to \Gamma$, define $\pi(P) \in \Gamma^{m \times n}$ by $\pi(P)_i^t = \pi(P_{[i\ell,(i+1)\ell-1]}^{[tk,(t+1)k-1]})$, and extend this to two-dimensional configurations. Define the *Cantor metric* on $\Sigma^{\mathbb{Z}}$ by

$$d_C(x,y) = \inf\{2^{-n} \mid n \in \mathbb{N}, x_{[-n,n]} = y_{[-n,n]}\}.$$

A *subshift* is a topologically closed set $X \subset \Sigma^{\mathbb{Z}}$ satisfying $\sigma(X) = X$, where $\sigma : \Sigma^{\mathbb{Z}} \to \Sigma^{\mathbb{Z}}$ is the *shift map*, defined by $\sigma(x)_i = x_{i+1}$ for all $x \in \Sigma^{\mathbb{Z}}$ and $i \in \mathbb{Z}$. The *language* of X, denoted by $\mathcal{B}(X)$, is the set of words in Σ^* that occur as a subpattern in some configuration of X. For $\ell \in \mathbb{N}$, we denote $\mathcal{B}_\ell(X) = \mathcal{B}(X) \cap \Sigma^\ell$.

A *cellular automaton* is a mapping $\Psi : X \to X$ on a subshift $X \subset \Sigma^{\mathbb{Z}}$ defined by a *local function* $\psi : \mathcal{B}_{2r+1}(X) \to \Sigma$ such that $\Psi(x)_i = \psi(x_{i-r}, \dots, x_{i+r})$ for all $x \in X$ and $i \in \mathbb{Z}$. Alternatively, cellular automata are exactly the d_C-continuous functions from X to itself that commute with the shift map σ [12]. Note that the shift map is a cellular automaton itself. Abusing notation, a cellular automaton can also be applied to a word $w \in \Sigma^n$ if $n \geq 2r$, the result being $\Psi(w) = \psi(w_{[0,2r]})\psi(w_{[1,2r+1]}) \cdots \psi(w_{[n-2r-1,n-1]}) \in \Sigma^{n-2r}$. A state $B \in \Sigma$ is *quiescent* for Ψ if $B^{2r+1} \in \mathcal{B}(X)$ and $\psi(B^{2r+1}) = B$. Unless otherwise noted, we only consider automata with $X = \Sigma^{\mathbb{Z}}$, $r = 1$ and state set $\Sigma = \{0,1\}^K$ for some $K \in \mathbb{N}$. In our constructions, we will partition the integer interval $[0, K-1]$ into disjoint subintervals $[k_0, k_1 - 1], [k_1, k_2 - 1], \dots, [k_{n-1}, k_n - 1]$, where $k_0 = 0$ and $k_n = K$. Then each state vector $v \in \{0,1\}^K$ can be expressed as $v = w_0 w_1 \cdots w_{n-1}$, where $w_i \in \{0,1\}^{k_{i+1}-k_i}$. The subintervals $[k_i, k_{i+1} - 1]$ are called *fields*, and the binary vectors w_i are the *values* of those fields for the state v. If a field is given a name, say $[k_i, k_{i+1} - 1] = \texttt{Field}_i$, then we denote $w_i = \texttt{Field}_i(v)$. We will also denote $\|\Psi\| = \|\Sigma\| = K$, and $B_\Sigma = 0^{\|\Sigma\|}$ (the *blank state* of Σ). *Unless otherwise noted, we always assume the blank state to be quiescent.*

Let $\Psi : \Sigma^{\mathbb{Z}} \to \Sigma^{\mathbb{Z}}$ be a CA with local function ψ. We define two *nondeterministic cellular automata* associated to Ψ, denoting the powerset of a set X by $\mathcal{P}(X)$. First, $\tilde{\Psi} : \Sigma^{\mathbb{Z}} \to \mathcal{P}(\Sigma^{\mathbb{Z}})$ is defined by

$$\tilde{\Psi}(x) = \{y \in \Sigma^{\mathbb{Z}} \mid \forall i \in \mathbb{Z} : y_i = \psi(x_{i-1}, x_i, x_{i+1}) \vee y_i = B_\Sigma\}.$$

Second, for all triples $(a,b,c) \in \Sigma^3$, define $f(a,b,c) \subset \Sigma^3$ as the set of triples obtained by setting a subset of the coordinates of (a,b,c) to the blank state B_Σ. Then define $\hat{\Psi} : \Sigma^{\mathbb{Z}} \to \mathcal{P}(\Sigma^{\mathbb{Z}})$ by

$$\hat{\Psi}(x) = \{y \in \Sigma^{\mathbb{Z}} \mid \forall i \in \mathbb{Z} : \exists v \in f(x_{i-1}, x_i, x_{i+1}) : y_i = \psi(v)\}.$$

Intuitively, $\tilde{\Psi}$ behaves like the corresponding deterministic automaton, but in any coordinate it may choose to produce the blank state B_Σ instead of the 'correct' value. The other nondeterministic automaton, $\hat{\Psi}$, has even more freedom, as it can choose to regard any cell in a local neighborhood as blank. These objects are not standard in the literature, but we will use them in the following way. The simulations of a CA Ψ by another CA Φ that we construct are not perfect, in the sense that not all initial configurations of Φ result in a correct simulation of Ψ. However, in our case they always result in the simulation of $\tilde{\Psi}$ or $\hat{\Psi}$, which are asymptotically at least as sparse as Ψ.

Let Ψ be a cellular automaton on $\Sigma^{\mathbb{Z}}$. We say that a two-dimensional configuration $\eta \in \Sigma^{\mathbb{Z}^2}$ (or $\Sigma^{\mathbb{Z}\times\mathbb{N}}$) is a (one-directional) *trajectory of* Ψ if $\eta^{t+1} = \Psi(\eta^t)$ for all $t \in \mathbb{Z}$ ($t \in \mathbb{N}$, respectively). The set of all (one-directional) trajectories of Ψ is denoted by $\mathcal{T}_\Psi$ ($\mathcal{T}^+_\Psi$). Trajectories of the associated nondeterministic automata are defined analogously, and the corresponding sets are denoted by $\tilde{\mathcal{T}}_\Psi$, $\tilde{\mathcal{T}}^+_\Psi$, $\hat{\mathcal{T}}_\Psi$ and $\hat{\mathcal{T}}^+_\Psi$. Clearly $\mathcal{T}_\Psi \subset \tilde{\mathcal{T}}_\Psi \subset \hat{\mathcal{T}}_\Psi$ holds, as does the analogous inclusion chain for one-directional trajectories. Note that each two-directional trajectory can also be seen as a one-directional trajectory by ignoring the cells with negative t-coordinate. The set $\Omega_\Psi = \bigcap_{n\in\mathbb{Z}} \Psi^n(\Sigma^{\mathbb{Z}})$ is called the *limit set* of Ψ. An alternative characterization for the limit set is $\Omega_\Psi = \{\eta^0 \mid \eta \in \mathcal{T}_\Psi\}$ [4].

A cellular automaton Ψ is called *nilpotent* if there exists $n \in \mathbb{N}$ such that $\Psi^n(x) = {}^\infty B^\infty_\Sigma$ for all $x \in \Sigma^{\mathbb{Z}}$. This is equivalent to the condition $\Omega_\Psi = \{{}^\infty B^\infty_\Sigma\}$.

A *topological dynamical system* is a tuple (X, T), where X is a compact metric space and $T : X \to X$ a continuous map. A probability measure μ on the Borel subsets of X is T*-invariant*, if $\mu(Y) = \mu(T^{-1}(Y))$ holds for all Borel sets $Y \subset X$. The set of T-invariant probability measures on X is denoted by $\mathcal{M}_T$. It is known that this set is always nonempty [20]. The map T, or the system (X, T), is called *uniquely ergodic* if $|\mathcal{M}_T| = 1$.

We define the *arithmetical hierarchy* of logical formulae over $\mathbb{N}$ as follows. A *bounded quantifier* has the form $\forall n < \tau$ or $\exists n < \tau$, where τ is a term which does not contain n (but may contain other free variables). A first-order arithmetical formula whose every quantifier is bounded has the classifications Π^0_0 and Σ^0_0. If ϕ is Σ^0_n (Π^0_n), then all formulae logically equivalent to $\forall n : \phi$ ($\exists n : \phi$) are Π^0_{n+1} (Σ^0_{n+1}, respectively). A set $X \subset \mathbb{N}$ gets the same classification as a formula ϕ, if $X = \{n \in \mathbb{N} \mid \phi(n)\}$ holds. It is known that the class of Σ^0_1-subsets of $\mathbb{N}$ consists of exactly the recursively enumerable sets, and Π^0_1 of their complements. When classifying sets of objects other than natural numbers, such as cellular automata, we assume that the objects are in a natural and computable bijection with $\mathbb{N}$. For a general introduction to the topic, see [16, Chapter IV.1].

For two sets $X, Y \subset \mathbb{N}$, we say that Y is *reducible to* X (also known as *many-one reducible*) if there exists a total computable function $f : \mathbb{N} \to \mathbb{N}$ such that $f(Y) \subset X$ and $f(\mathbb{N} \setminus Y) \subset \mathbb{N} \setminus X$. For a class Ξ of subsets, we say that X is Ξ*-hard* if every Y in Ξ is reducible to X. If X also lies in Ξ, we say that X is Ξ*-complete*. Note that the famous class of *NP-complete problems* does not fit into this framework, since it is defined using the stronger notion of *polynomial reducibility*, which further assumes that the function f is computable in polynomial time.

The operator mod has two related but completely orthogonal uses in this paper. First, if $i, j \in \mathbb{Z}$ and $n \in \mathbb{N}$, the notation $i \equiv j \bmod n$ is equivalent to the formula $\exists k \in \mathbb{Z} : i - j = kn$. Here, mod is used in conjunction with $\equiv$ to form a ternary predicate. Second, if $i \in \mathbb{Z}$ and $n \in \mathbb{N}$, then $i \bmod n$ denotes the unique number $j \in [0, n-1]$ with $i \equiv j \bmod n$. In this case, mod is a binary function from $\mathbb{Z} \times \mathbb{N}$ to $\mathbb{N}$.

This paper contains several pictures of spacetime diagrams of cellular automata. As some of the automata contain Turing machine computations, and it is customary to illustrate these using a time axis that increases upwards, we have chosen this representation also for our spacetime diagrams.

3. Preliminary Results

In this section, we state some preliminary results about nilpotency and unique ergodicity in cellular automata. We begin with a characterization of nilpotency. The first and third claims are proved in [4], and the second in [9]. We do not use this result explicitly, and it is meant to be a formal version of the remarks on nilpotency in Section 1.

Proposition 3.1. *Let $\Psi : \Sigma^{\mathbb{Z}} \to \Sigma^{\mathbb{Z}}$ be a cellular automaton with quiescent state B_Σ. The following conditions are equivalent to the nilpotency of Ψ.*

- *For all $x \in \Sigma^{\mathbb{Z}}$, there exists $n \in \mathbb{N}$ such that $\Psi^n(x) = {}^\infty B_\Sigma^\infty$.*
- *For all $x \in \Sigma^{\mathbb{Z}}$ and $i \in \mathbb{Z}$, there exists $n \in \mathbb{N}$ such that $\Psi^m(x)_i = B_\Sigma$ for all $m \geq n$.*
- *The limit set of Ψ is a singleton: $\Omega_\Psi = \{{}^\infty B_\Sigma^\infty\}$.*

Next, let us consider unique ergodicity. Of course, a nilpotent cellular automaton $\Psi : \Sigma^{\mathbb{Z}} \to \Sigma^{\mathbb{Z}}$ is uniquely ergodic, since $\mathcal{M}_\Psi$ consists of the Dirac measure μ_0 defined by

$$\mu_0(Y) = \begin{cases} 1, & \text{if } {}^\infty B_\Sigma^\infty \in Y \\ 0, & \text{if } {}^\infty B_\Sigma^\infty \notin Y. \end{cases} \tag{1}$$

General invariant measures can be tricky to work with, so we prove a combinatorial characterization (Corollary 3.1) of unique ergodicity for cellular automata with a quiescent state. Uniquely ergodic cellular automata turn out to have a strong nilpotency-like property. First, we show that it suffices to consider the limit set.

Lemma 3.1. *A cellular automaton $\Psi : \Sigma^{\mathbb{Z}} \to \Sigma^{\mathbb{Z}}$ is uniquely ergodic if and only if its restriction $\Psi|_{\Omega_\Psi}$ to the limit set is.*

Proof. Denote $\Phi = \Psi|_{\Omega_\Psi}$. If Ψ is uniquely ergodic, then clearly Φ must also be, since any invariant measure of the latter is also in $\mathcal{M}_\Psi$.

Let then $\mu \in \mathcal{M}_\Psi$ be arbitrary. We will prove that $\mu(\Omega_\Psi) = 1$, showing that the restriction of μ to subsets of Ω_Ψ is in $\mathcal{M}_\Phi$. For that, let

$$X_n = \{x \in \Sigma^{\mathbb{Z}} \mid \Psi^{-n}(x) \neq \emptyset, \Psi^{-n-1}(x) = \emptyset\} \subset \Sigma^{\mathbb{Z}}$$

for all $n \in \mathbb{N}$. Clearly, the sets X_n are pairwise disjoint, and cover the set $\Sigma^{\mathbb{Z}} - \Omega_\Psi$. We also have that $\mu(X_n) = \mu(\Psi^{-n-1}(X_n)) = 0$ for all n, and it follows that $\mu(\Omega_\Psi) = \mu(\Sigma^{\mathbb{Z}}) - \mu(\bigcup_{n\in\mathbb{N}} X_n) = 1$. Thus if Ψ is not uniquely ergodic, then neither is Φ. □

We need the following classical result of ergodic theory:

Lemma 3.2 (Theorem 6.19 of [20])**.** *Let (X, T) be a topological dynamical system. The following conditions are equivalent:*

1. *The map T is uniquely ergodic.*
2. *There exists a measure $\mu \in \mathcal{M}_T$ such that $(\frac{1}{n}\sum_{i=0}^{n-1} f(T^i(x)))_{n\in\mathbb{N}}$ converges to $\int_X f\, d\mu$ for every $x \in X$ and continuous $f : X \to \mathbb{C}$.*
3. *For every continuous $f : X \to \mathbb{C}$, the sequence $(\frac{1}{n}\sum_{i=0}^{n-1} f \circ T^i)_{n\in\mathbb{N}}$ converges uniformly to a constant function.*

For the next result and its proof, we make the following definition.

Definition 3.1. Let $X \subset \Sigma^{\mathbb{Z}}$ be a subshift, let $\Psi : X \to X$ be a cellular automaton, let $w \in \mathcal{B}_{2r+1}(X)$, and let $x \in X$. We denote

$$d_\Psi(w, x) = \liminf_{n\to\infty} \frac{1}{n} |\{t \in [0, n-1] \mid \Psi^t(x)_{[-r,r]} = w\}|,$$

the lower asymptotic density of the occurrences of the word w at the origin.

The combinatorial characterization of uniquely ergodic cellular automata is the following.

Proposition 3.2. *Let $X \subset \Sigma^{\mathbb{Z}}$ be a subshift with ${}^\infty B_\Sigma^\infty \in X$, and let $\Psi : X \to X$ be a cellular automaton. Then Ψ is uniquely ergodic if and only if $d_\Psi(B_\Sigma, x) = 1$ holds for all $x \in X$. If this is the case, the convergence of the limit is uniform in x.*

Proof. Suppose first that $d_\Psi(B_\Sigma, x) = 1$ holds for all $x \in X$. Let μ_0 be the Dirac probability measure on X defined by (1). Then μ_0 is Ψ-invariant, since B_Σ is quiescent, and $\int_X f\, d\mu_0 = f({}^\infty B_\Sigma{}^\infty)$ holds for all continuous functions $f : X \to \mathbb{C}$.

Let $x \in X$ and $f : X \to \mathbb{C}$ continuous. For each $n \in \mathbb{N}$, let $f_n : X \to \mathbb{C}$ be a (continuous) function of the form $\sum_{i=1}^k a_i \mathbf{1}_{C_i}$ with $a_i \in \mathbb{C}$ and $C_i \subset X$ a cylinder set (a set of the form $\{y \in X \mid y_{[-r,r]} = w\}$ for a word $w \in \mathcal{B}_{2r+1}(X)$), and such that $|f(y) - f_n(y)| < \frac{1}{n}$ for all $y \in X$. This is possible, since functions of this form are dense in the uniform topology.

The condition $d_\Psi(B_\Sigma, x) = 1$ implies that $d_\Psi(B_\Sigma^{2r+1}, x) = 1$ for all $r \in \mathbb{N}$, and thus for all $n \in \mathbb{N}$, the average $\frac{1}{m}\sum_{i=0}^{m-1} f_n(\Psi^i(x))$ converges to the constant

$f_n(^\infty B_\Sigma{}^\infty)$ as m grows. Hence also $\frac{1}{m}\sum_{i=0}^{m-1} f(\Psi^i(x))$ converges to $f(^\infty B_\Sigma{}^\infty)$, which can be proved using a standard $\frac{\epsilon}{3}$-approximation. Unique ergodicity now follows by Lemma 3.2.

Let then Ψ be uniquely ergodic, and consider the (again continuous) function $f : \Sigma^{\mathbb{Z}} \to \{0, 1\}$ defined by $f^{-1}(1) = \{y \in \Sigma^{\mathbb{Z}} \mid y_0 = B_\Sigma\}$. By Lemma 3.2, the sequence $(\frac{1}{n}\sum_{i=0}^{n-1} f \circ \Psi^i)_{n\in\mathbb{N}}$ converges uniformly to a constant, which must be 1 because of the fixed point $^\infty B_\Sigma{}^\infty$. This implies that $d_\Psi(B_\Sigma, x) = 1$, and that the convergence is uniform in x. □

Lemma 3.1 and the above together imply the following.

Corollary 3.1. *A cellular automaton $\Psi : \Sigma^{\mathbb{Z}} \to \Sigma^{\mathbb{Z}}$ is uniquely ergodic if and only if $d_\Psi(B_\Sigma, x) = 1$ holds for all $x \in \Omega_\Psi$, and then the convergence of the limit is uniform in x.*

Another way to state Corollary 3.1 is that a CA $\Psi : \Sigma^{\mathbb{Z}} \to \Sigma^{\mathbb{Z}}$ is uniquely ergodic if and only if the asymptotic density of B_Σ-cells in the central column $\eta_0^{[0,\infty)}$ is 1 for every $\eta \in \mathcal{T}_\Psi$. Since this then holds for all columns, the automaton Ψ acts in a very 'sparse' environment, forming large areas of blank cells almost everywhere.

4. Simulation

In this section, we define the different notions of simulation between cellular automata that we use in the course of this paper. We begin with the definition of basic simulation, which is somewhat nonstandard, since it is specifically tailored to our needs.

Definition 4.1. Let $\Psi : \Sigma^{\mathbb{Z}} \to \Sigma^{\mathbb{Z}}$ and $\Phi : \Gamma^{\mathbb{Z}} \to \Gamma^{\mathbb{Z}}$ be two cellular automata. A *simulation* of Φ by Ψ is a quintuple $(Q, U, (a, b), C, \pi)$, where $Q, U \in \mathbb{N}$ are the *dimensions* of the simulation, $(a, b) \in [0, Q-1] \times [0, U-1]$ is the *base coordinate*, $C \subset \Sigma^{Q\times U}$ is the set of *macro-cells* and $\pi : \Sigma^{Q\times U} \to \Gamma$ is the *state function*, such that $\pi(\Sigma^{Q\times U} \setminus C) = B_\Gamma$ and $\mathcal{T}_\Phi^+ \subset \pi(\mathcal{T}_\Psi^+)$ hold, as does $P_a^b \neq B_\Sigma$ for all $P \in C$. If such a simulation exists, we say that Ψ (Q, U)-*simulates* Φ. The base coordinate will be dropped from the definition if it is unnecessary or clear from the context.

An *amplifier* is a sequence $(\Psi_n)_{n\in\mathbb{N}}$ of cellular automata such that for each $n \in \mathbb{N}$, the automaton Ψ_n simulates Ψ_{n+1}. The *bottom* of the amplifier is the automaton Ψ_0.

A simulation is a method of constructing, for each trajectory of Φ, a corresponding trajectory of Ψ, where the correspondence is given by a function that associates a state of Φ to every $Q \times U$-rectangle of Ψ-states. An amplifier, especially one where the Ψ_n are somewhat similar, is a form of *approximate self-simulation*. The term amplifier is also used in [7] with a similar meaning, while the notion of simulation in said paper is much more general. Note that the base coordinate (a, b) and the set C of macro-cells are largely irrelevant in

the definition. Next, we define some more refined classes of simulations that depend more heavily on (a, b) and C.

Definition 4.2. Let (Q, U, C, π) be a simulation of $\Phi : \Gamma^{\mathbb{Z}} \to \Gamma^{\mathbb{Z}}$ by $\Psi : \Sigma^{\mathbb{Z}} \to \Sigma^{\mathbb{Z}}$. We say that the simulation is *rigid*, if $\pi(\tilde{\mathcal{T}}_\Psi) \subset \tilde{\mathcal{T}}_\Phi$, *weakly rigid*, if $\pi(\tilde{\mathcal{T}}_\Psi) \subset \hat{\mathcal{T}}_\Phi$, and *strongly rigid*, if $\pi(\hat{\mathcal{T}}_\Psi) \subset \tilde{\mathcal{T}}_\Phi$.

Intuitively, a simulation is rigid if non-blank macro-cells cannot appear from nothing, but must be the result of locally correct simulation by other macro-cells, in nondeterministic two-directional trajectories. Weak and strong rigidity are technical variations of the concept. We will only construct simulations that are at least weakly rigid, since they are much easier to reason about than general simulations, where the π-image of a trajectory of Ψ may not be related to Φ in any way. In our case, even the trajectories of the nondeterministic CA $\tilde{\Phi}$ and $\hat{\Phi}$ will be sparse, and a rigid simulation transfers this sparsity to Ψ. In the course of the proof, we construct an amplifier $(\Psi_n)_{n\in\mathbb{N}}$ with rigid simulations, where each $\tilde{\Psi}_n$ is sparser than the previous one. Since this sparsity is transferred to the bottom automaton Ψ_0 as described above, it is uniquely ergodic.

Definition 4.3. Let $\Psi : \Sigma^{\mathbb{Z}} \to \Sigma^{Z}$ be a CA with local function ψ, and let $\delta \in \{-1, 0, 1\}$. A state $c \in \Sigma$ is called *δ-demanding*, if $\psi(c_{-1}, c_0, c_1) = c$ implies $c_\delta \neq B_\Sigma$ for all $c_{-1}, c_0, c_1 \in \Sigma$. We assign two directed graphs G and G' to every configuration $\eta \in \Sigma^{\mathbb{Z}^2}$ as follows. The vertex set of both of them consists of the non-blank cells of η. There is an edge in G' from the cell $\eta_{i+\delta}^{t-1}$ to η_i^t, where $i, t \in \mathbb{Z}$ and $\delta \in \{-1, 0, 1\}$, if the cells are not blank, and in G if η_i^t is also δ-demanding.

Cells connected by an edge in G (G') are called *(weakly) adjacent*. Two cells are *(directly) connected* if there is a (directed) path in G between them. The analogous weak concepts are defined for G'.

Let $(Q, U, (a, b), C, \pi)$ be a simulation of $\Phi : \Gamma^{\mathbb{Z}} \to \Gamma^{\mathbb{Z}}$ by Ψ. We say that the simulation is *connecting*, if for all trajectories $\eta \in \tilde{\mathcal{T}}_\Psi$ and all $i, t \in \mathbb{Z}$ and $\delta \in \{-1, 0, 1\}$ such that $\pi(\eta)_i^t$ and $\pi(\eta)_{i+\delta}^{t-1}$ are adjacent, the base cells η_{iQ+a}^{tU+b} and η_{jQ+a}^{sU+b} are directly connected. The simulation is *strongly connecting*, if the above holds for all $\eta \in \hat{\mathcal{T}}_\Psi$.

Intuitively, a connecting simulation respects the existence of directed paths. See Figure 1 for a visualization of a connecting simulation. The concepts of adjacent and connected cells are somewhat technical, since two neighboring non-blank cells may not be even indirectly connected, but they simplify our constructions and proofs. For this reason, they are used more frequently than their more natural weak counterparts. In particular, connecting simulations with lots of demanding cells enable certain geometric arguments, where we show that the bases of two macro-cells in a trajectory must both be directly connected to some third cell, which implies that the macro-cells are correctly aligned or share some other desired property.

We now show that the simulation relation is transitive.

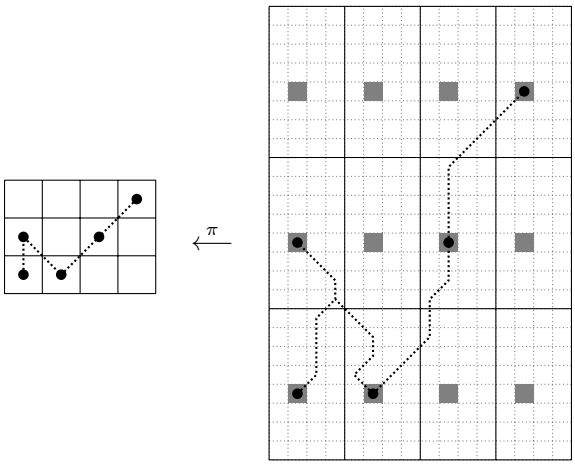

Figure 1: A connecting simulation. The connecting directed paths on the right corresponds to the adjacencies in the π-image on the right. The base cells of the macro-cells are shaded.

Lemma 4.1. *Let $\Psi : \Sigma^{\mathbb{Z}} \to \Sigma^{\mathbb{Z}}$, $\Phi : \Gamma^{\mathbb{Z}} \to \Gamma^{\mathbb{Z}}$ and $\Theta : \Delta^{\mathbb{Z}} \to \Delta^{\mathbb{Z}}$ be three cellular automata, and suppose that Ψ (Q, U)-simulates Φ, and Φ (Q', U')-simulates Θ. Then, Ψ (QQ', UU')-simulates Θ.*

Proof. Denote by $S = (Q, U, (a, b), C, \pi)$ the simulation of Φ by Ψ, and by $S' = (Q', U', (a', b'), C', \pi')$ the simulation of Θ by Φ. We define the set $C'' \subset \Sigma^{QQ' \times UU'}$ of macro-cells for the composition simulation by $C'' = \pi^{-1}(C')$, that is, a pattern $P \in \Sigma^{QQ' \times UU'}$ is in C'' if and only if $\pi(P) \in C'$. The state function $\pi'' : \Sigma^{QQ' \times UU'} \to \Delta$ is the composition of π' and π. Define also $v = (a + Qa', b + Ub')$. Then $(QQ', UU', v, C'', \pi'')$ is easily seen to be a simulation of Θ by Ψ. □

The simulation $(QQ', UU', v, C'', \pi'')$ of Θ by Ψ constructed above is called the *composition* of S and S', denoted $S' \circ S$. The order of composition here is chosen to correspond to the composition of the state functions.

Lemma 4.2. *Retain the assumptions and notation of Lemma 4.1. If S and S' are rigid and connecting, then so is $S' \circ S$. If S is weakly rigid and connecting, and S' is strongly rigid and strongly connecting, then $S' \circ S$ is again rigid and connecting.*

Proof. First, if S and S' are rigid, we have $\pi''(\tilde{\mathcal{T}}_\Psi) = \pi'(\pi(\tilde{\mathcal{T}}_\Psi)) \subset \pi'(\tilde{\mathcal{T}}_\Phi) \subset \tilde{\mathcal{T}}_\Theta$, and then the composition is rigid.

Let S and S' be also connecting, and let $\eta \in \tilde{\mathcal{T}}_\Psi$. If two cells $\pi'(\pi(\eta))_i^t$ and $\pi'(\pi(\eta))_j^s$ are adjacent, the bases $\pi(\eta)_{iQ'+a'}^{tU'+b'}$ and $\pi(\eta)_{jQ'+a'}^{sU'+b'}$ of the macro-cells $\pi(\eta)_{[iQ',(i+1)Q'-1]}^{[tU',(t+1)U'-1]}$ and $\pi(\eta)_{[jQ',(j+1)Q'-1]}^{[sU',(s+1)U'-1]}$ of S' are directly connected, since S' is connecting. The directed path $(\pi(\eta)_{k_m}^{r_m})_{m=0}^{M}$ between the bases, where $(k_0, r_0) =$

$(iQ'+a', tU'+b')$ and $(k_M, r_M) = (jQ'+a', sU'+b')$, consists of adjacent cells, and since $\pi(\eta) \in \tilde{\mathcal{T}}_\Phi$ by rigidity and S is connecting, the bases $\eta_{k_mQ+a}^{r_mU+b}$ and $\eta_{k_{m+1}Q+a}^{r_{m+1}U+b}$ of the macro-cells $\eta_{[k_mQ,(k_m+1)Q-1]}^{[r_mU,(r_m+1)U-1]}$ and $\eta_{[k_{m+1}Q,(k_{m+1}+1)Q-1]}^{[r_{m+1}U,(r_{m+1}+1)U-1]}$ of S are directly connected for all $m \in [0, M-1]$. But then the bases $\eta_{(iQ'+a')Q+a}^{(tU'+b')U+b}$ and $\eta_{(jQ'+a')Q+a}^{(sU'+b')U+b}$ of the macro-cells $\eta_{[iQQ',(i+1)QQ'-1]}^{[tUU',(t+1)UU'-1]}$ and $\eta_{[jQQ',(j+1)QQ'-1]}^{[sUU',(s+1)UU'-1]}$ of $S' \circ S$ are directly connected, and thus the composition is connecting.

The second claim is completely analogous, simply replace $\tilde{\mathcal{T}}_\Phi$ by $\hat{\mathcal{T}}_\Phi$ in the above. □

5. Cellular Automata Represented by Programs

We now introduce a programming language for expressing CA rules. As in [7], we use it both for precisely describing complicated CA rules, and for encoding said rules into short bitstrings which can then be interpreted by Turing machines, in order to perform programmatic simulation. The language describes automata whose states are binary vectors divided into fields, as defined in Section 2. We do not rigorously define this language, but go through its relevant aspects using an example automaton, presented as a program in Algorithm 1. The program presented here is designed for illustrative purposes, and does not exhibit much interesting behavior. As a convention, we only display line numbers in algorithms if we explicitly refer to them in the main text.

Algorithm 1 An example program of a cellular automaton.

```
 1: num param Modulus = 10
 2: num field Counter ≤ Modulus − 1
 3:
 4: procedure PROC(n)
 5:     if Counter⁻ = n then
 6:         Counter ← Counter + 1 mod Modulus
 7:
 8: if Counter ≠ Counter⁺ then
 9:     PROC(Counter⁺)
10: else
11:     PROC(0)
```

A program of the language consists of zero or more *parameter and field definitions* in any order, followed by one or more *procedure definitions* and finally the *program body*. On line 1.1, an integer parameter called `Modulus` is defined. Parameters should be viewed as constants, as their value is the same in all cells of the automaton, at all times.

Likewise, line 1.2 defines an integer field with maximum value $\texttt{Modulus} - 1 = 9$ (stored as a bitstring of length $\lceil \log_2 9 \rceil = 4$). As opposed to parameters, fields make up the state of a cell, and their value typically changes as the cellular automaton is applied. In this example, the state set of the automaton is

$\{0,1\}^4$, and it consists of this single field. Fields and parameters can also contain Boolean truth values (stored as a single bit, with 0 = `False` and 1 = `True`), bitstrings of some fixed length, or enumerations, which are simply integers with special labels. For technical reasons that we explain later in this section, the length of a bitstring field must be given in unary notation.

Line 1.4 marks the beginning of a procedure definition. The name of the procedure is PROC, and it takes one argument, n. Line 1.5 marks the beginning of a conditional block, which is executed if the value of the implicitly defined parameter `Counter`$^-$, which contains the value of the `Counter` field of the left neighbor, is exactly n. Its counterpart `Counter`$^+$, containing the value of the `Counter` field of the left neighbor, is referenced in line 1.8. Only fields can be assigned new values; parameters are read-only, as are the fields of the neighboring cells. On line 1.6, the `Counter` field of the cell is assigned a new value: it is incremented by one modulo the parameter `Modulus`.

The body of the program contains another conditional block, with two calls to the procedure PROC on lines 1.9 and 1.11, with the respective arguments `Counter`$^+$ and 0. The transition rule of the CA consists of executing the program on every cell of the configuration. In this case, the program encodes (in a somewhat convoluted way) the local rule

$$(a,b,c) \mapsto \begin{cases} (b+1) \bmod 10, & \text{if } b = c = 0 \text{ or } b \neq c = a \\ b, & \text{otherwise,} \end{cases}$$

where $a, b, c \in \{0, \ldots, 2^4 - 1 = 15\}$.

The programming language also contains a special keyword `This`. It is handled as an implicitly defined bitstring parameter, whose value is *the whole program*, using some suitable encoding. In this example, the value of `This` would be the whole of Algorithm 1, encoded as a bitstring. It gives every program written in the language the ability to easily examine itself, and is mainly used for self-simulation. We remark here that there are multiple ways of implementing programmatic self-simulation, but all of them include some 'trick' that gives the cellular automaton access to its own definition. We have chosen the approach of allowing our programs to explicitly refer to themselves, with the goal of being as easy to understand as possible.

This example contains most of the basic constructs of the programming language. Procedure calls can appear in the bodies of other procedures, but recursion is explicitly prohibited, so there may not be a cycle of procedures with each calling the next. The language can be assumed to contain any particular polynomial-time arithmetic and string manipulation functions we need, the latter of which can be used also on numerical fields. It is clear that all cellular automata can be described by a program, if only by listing their entire transition table in a series of nested conditional blocks. Conversely, we say a bitstring p is a *valid program* if it defines a cellular automaton.

We may now construct a Turing machine Int, the *interpreter*, that takes as input four binary words r, s, t, p and outputs $\phi(r, s, t)$, where $\phi : \Sigma^3 \to \Sigma$ is the local function of the automaton $\Psi : \Sigma^{\mathbb{Z}} \to \Sigma^{\mathbb{Z}}$ described by the valid program

p. In this case, we denote $\phi = \mathrm{Int}_p$, $\Psi = \mathrm{Int}_p^\infty$ and $\|\Sigma\| = \|p\|$ (not to be confused with the length of the program p as a bitstring, which is denoted by $|p|$). Since we explicitly prohibited recursion, no form of looping is available, the length of each field is proportional to the length of its definition (here we need the unary encoding of bitstring lengths), and all functions run in polynomial time, it is clear that Int has time complexity $P^T_{\mathrm{Int}}(|p|, \|p\|)$ and space complexity $P^S_{\mathrm{Int}}(|p|, \|p\|)$, where P^T_{Int} and P^S_{Int} are some polynomials in two variables with integer coefficients.

Finally, we define CA transformations, which are functions that modify programmatically defined cellular automata.

Definition 5.1. Let $G : \mathbb{N} \times \{0,1\}^* \to \{0,1\}^*$ be a function. We say that G is a *polynomial CA transformation*, if the following conditions hold.

- $G(n,p)$ is computable in time polynomial in n and $|p|$.
- If p is a valid program, then so is $G(n,p)$ for all $n \in \mathbb{N}$.
- If p is a valid program, then $\|G(n,p)\|$ is polynomial in n and $\|p\|$.

If the automaton $\mathrm{Int}^\infty_{G(n,p)}$ only depends on Int^∞_p for all $n \in \mathbb{N}$ and all valid programs p, we say G is *consistent*, and write $G(n,\Psi)$ for the automaton $\mathrm{Int}^\infty_{G(n,p)}$, if Ψ is defined by the program p. If G and G' are two polynomial CA transformations, so is the function $(n,p) \mapsto G(n, G'(n,p)))$, which is called their *composition* and denoted $G \circ G'$.

6. Sparsification

6.1. The Sparsification Transformation

In this section, we define a specific polynomial CA transformation G_s that performs *sparsification*, and for that, let $\Phi = \mathrm{Int}^\infty_p : \Gamma^{\mathbb{Z}} \to \Gamma^{\mathbb{Z}}$ be a cellular automaton defined by the program $p \in \{0,1\}^*$, and let $n \in \mathbb{N}$ be arbitrary. Intuitively, the CA defined by $G_s(n,p)$ behaves exactly like Φ, except that all of its trajectories will eventually become *sparse*, that is, they consist mostly of cells in the blank state. In fact, the transformed automaton $\mathrm{Int}^\infty_{G_s(n,p)}$ simulates Φ with macro-cells that consist mostly of blank cells.

The transformation G_s works as follows. First, split the program p into two parts p_{def} and p_{body}, the first of which contains all the parameter, field and procedure definitions, and the second the main program body. For a fixed $n \in \mathbb{N}$, the output of $G_s(n,p)$ is then Algorithm 2, along with the definitions of the procedures Die, Blank, InputTriple, CopyRight and CopyLeft. If the original program happens to contain a parameter, field or procedure whose name clashes with the ones defined here, it will be renamed, together with all references to it. It is clear that the conditions for being a consistent polynomial CA transformation are fulfilled.

We now explain the new fields and their functionality in the sparsified automaton. Denote by Σ the state set of $\Psi = \mathrm{Int}^\infty_{G_s(n,p)}$. The boolean field `Live`,

Algorithm 2 The sparsified program $G_s(n, p)$, minus the new procedures.

num param $\texttt{N} = n$ ▷ The level of sparsification
bool field `Live` ▷ Marks cells that are not blank
enum field $\texttt{Kind} \in \{\Leftarrow, \Rightarrow, \Leftrightarrow\}$ ▷ Direction of movement
num field $\texttt{Cnt} \leq 2\,\texttt{N}$ ▷ Synchronizes the simulation
p_{def} ▷ The parameter, field and procedure definitions of p

if InputTriple **then**
    p_{body} ▷ Execute the transition function defined by p
    if Blank **then** Die
    else
        $\texttt{Live} \leftarrow \texttt{True}$
        $\texttt{Cnt} \leftarrow 0$
        $\texttt{Kind} \leftarrow \Rightarrow$
else if $\neg\texttt{Live} \wedge \neg\texttt{Live}^+ \wedge \texttt{Kind}^- = \Rightarrow \wedge\, \texttt{Cnt}^- < 2\,\texttt{N}$ **then**
    CopyLeft($\Rightarrow$)
else if $\texttt{Kind} = \Leftarrow \wedge \neg\texttt{Live}^+ \wedge \texttt{Kind}^- = \Rightarrow \wedge\, \texttt{Cnt}^- = \texttt{Cnt} = \texttt{N}$ **then**
    CopyLeft($\Rightarrow$)
else if $\neg\texttt{Live} \wedge \neg\texttt{Live}^- \wedge \texttt{Kind}^+ = \Rightarrow \wedge\, \texttt{Cnt}^+ = 0$ **then**
    CopyRight($\Leftarrow$)
else if $\neg\texttt{Live} \wedge \neg\texttt{Live}^- \wedge \texttt{Kind}^+ = \Leftarrow \wedge\, 0 < \texttt{Cnt}^+ < 2\,\texttt{N}$ **then**
    CopyRight($\Leftarrow$)
else if $\texttt{Kind} = \Rightarrow \wedge\, \neg\texttt{Live}^- \wedge \texttt{Kind}^+ = \Leftarrow \wedge\, \texttt{Cnt}^+ = \texttt{Cnt} = \texttt{N}$ **then**
    CopyRight($\Leftarrow$)
else if $\texttt{Kind} = \Leftrightarrow \wedge\, \neg\texttt{Live}^- \wedge \texttt{Kind}^+ = \Leftarrow \wedge\, \texttt{N} < \texttt{Cnt}^+ = \texttt{Cnt} < 2\,\texttt{N}$ **then**
    CopyRight($\Leftarrow$)
else if $\neg\texttt{Live} \wedge \neg\texttt{Live}^- \wedge \texttt{Kind}^+ = \Rightarrow \wedge\, \texttt{Cnt}^+ = \texttt{N}$ **then**
    CopyRight($\Leftrightarrow$)
else if $\neg\texttt{Live} \wedge \neg\texttt{Live}^- \wedge \texttt{Kind}^+ = \Leftrightarrow \wedge\, \texttt{N} < \texttt{Cnt}^+ < 2\,\texttt{N}$ **then**
    CopyRight($\Leftrightarrow$)
else Die

called the *live flag*, is used to indicate whether a cell is in the blank state B_Σ (`Live = False`) or not (`Live = True`). The field `Kind` specifies the *kind* of a non-blank cell. As there are three possible values for the field, so are there three possible kinds: *left moving*, *right moving* and *returning*, corresponding to the respective values $\Leftarrow$, $\Rightarrow$ and $\Leftrightarrow$. The cells of different kinds travel back and forth (returning cells travel to the left), exchanging information with each other in order to simulate Φ. The field `Cnt`, called the *counter*, is used to synchronize the simulation. Its maximum value is twice the parameter $\mathtt{N} = n$.

The procedure definitions and program body of $G_s(n, p)$ are independent of n. The procedure DIE makes the current cell assume the blank state $B_\Sigma = 0^{\|\Sigma\|}$. The procedure BLANK returns `True` if and only if the Γ-state of the current cell is B_Γ. The procedure INPUTTRIPLE returns `True` if and only if the cells in the neighborhood, if they are live, have counter value $2\,\mathtt{N}$, their kinds (if live) are $\Rightarrow$, $\Leftrightarrow$ and $\Leftarrow$, and at least one of them is live. Finally, COPYRIGHT, when given an argument $k \in \{\Leftarrow, \Rightarrow, \Leftrightarrow\}$, makes the current cell live, sets its kind to k, copies the counter value and Γ-state of the right neighbor, and then increments the counter value by one. The behavior of COPYLEFT is symmetric.

A correct simulation of Ψ proceeds as follows. In the beginning of what is called a *cycle* of $2n + 1$ steps, a right moving cell with counter value 0 is placed at the coordinate $i(2n + 1)$ for every $i \in \mathbb{N}$, with blank cells between them. In general, at each timestep $t \in \mathbb{N}$, every live cell will have counter value $t \bmod 2\,\mathtt{N} + 1$. The right moving cells create a left moving cell on their left, which has the same Σ-state as they do (line 2.19). These cells move in their respective directions until their counter values reach $2\,\mathtt{N}$, and the right and left cells originating from neighboring segments may cross at $t = \mathtt{N}$ (lines 2.17 and 2.23). At that time, the right moving cells also create a returning cell on their left (line 2.27), which starts moving left along the left moving cell. At $t = 2\,\mathtt{N}$, each coordinate $i(2n + 1)$ contains a returning cell with the Σ-state it had at the beginning of the cycle, and is surrounded by a right-moving and left-moving cell, containing the Σ-states of its left and right neighbors, respectively. The procedure INPUTTRIPLE returns `True` for the returning cells, and these three-cell groups are used to calculate new Γ-states for the cells, as per the rule of Φ (line 2.8). Note that if some of the cells in the neighborhood are blank, the computation on line 2.8 will be carried out as if they were live cells with Γ-state B_Γ. Also, if the Γ-state of the resulting cell is B_Γ, it will die. Thus the blank state B_Γ is simulated by the absence of a live cell.

See Figure 2 for a sparsification of the three-neighbor XOR automaton, which should clarify the above explanation.

We also slightly generalize the sparsification construction by defining partial sparsifications.

Definition 6.1. A *partial sparsification transformation* is a polynomial CA transformation G_s^N, where $N \subseteq \mathbb{N}$ is decidable in polynomial time, defined by

$$G_s^N(n, p) = \begin{cases} G_s(n, p), & \text{if } n \in N \\ p, & \text{otherwise.} \end{cases}$$

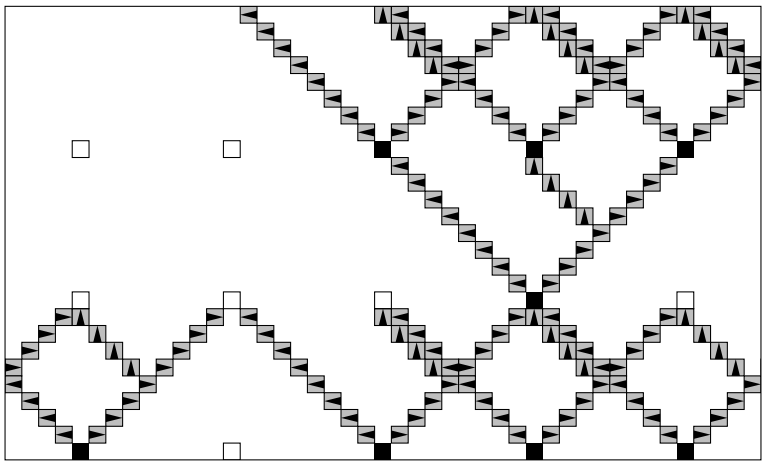

Figure 2: The sparsification transformation applied to the three-neighbor XOR automaton, with $n = 4$. The cells with left, right and up arrows are left moving, right moving and returning, respectively. The black cells are right moving with $\texttt{Cnt} = 0$, and white cells are blank. The live cells on each row have the same counter value.

Clearly, each G_s^N is also consistent. The set N is called the *effective set* of G_s^N. Partial sparsification transformations are used in the computability results of Section 10.3, where classes of cellular automata are separated by whether N is finite or infinite.

6.2. Properties of Sparsification

We now prove some basic properties of the sparsification transformation, keeping the notation of Section 6.1. For the rest of this section, fix $n \in \mathbb{N}$ and denote $M = 2n + 1$. First, there is the claim that the sparsified automaton actually simulates the original.

Lemma 6.1. *The CA* $\Psi = \mathrm{Int}^{\infty}_{G_s(n,p)}$ (M, M)*-simulates* Φ*.*

Proof. We define the set $C \subset \Sigma^{M^2}$ of macro-cells as the set of those rectangular patterns $P \in \Sigma^{M^2}$ that satisfy $\texttt{Live}(P_0^0) = \texttt{True}$, $\texttt{Kind}(P_0^0) = \texttt{Right}$ and $\texttt{Cnt}(P_0^0) = 0$. These are the black cells in Figure 2. For such a pattern P, define $\pi(P)$ simply as the Σ-state of P_0^0. If $P \notin C$, we of course define $\pi(P) = B_\Sigma$. We now claim that $(M, M, (0, 0), C, \pi)$ is the desired simulation.

Let thus $\eta \in \mathcal{T}_\Phi^+$ be a trajectory of Φ, and define $\xi \in \mathcal{T}_\Psi^+$ by $\xi^0_{[Qi,Q(i+1)-1]} = c_i^t B_\Sigma^{2n}$, where $c_i^t \in \Sigma$ is either a live right moving cell with counter value 0 and Γ-state η_i^0 if $\eta_i^t \neq B_\Gamma$, or B_Σ if $\eta_i^t = B_\Gamma$. Every other row of ξ is of course the Ψ-image of the previous row. Then we have $\pi(\xi)^0 = \eta^0$ by definition of π. The general case of $\pi(\xi)^t = \eta^t$ for all $t \in \mathbb{N}$ is easily verified by induction, using the rules of Ψ and the explanation of the simulation in the previous subsection. □

The simulation $(M, M, (0,0), C, \pi)$ defined above is called the *level-n sparse simulation of* Ψ. In general, the sparse simulation is not rigid, but we can quite easily show the following.

Lemma 6.2. *The sparse simulation of* Ψ *by* $\Phi = \mathrm{Int}^{\infty}_{G_s(n,p)}$ *is weakly rigid.*

Proof. Let $\eta \in \tilde{\mathcal{T}}_\Psi$ be arbitrary. It suffices to prove that we have $\pi(\eta)^1_0 = \mathrm{Int}_p(c_{-1}, c_0, c_1)$ in the case that $\pi(\eta)^1_0 \neq B_\Sigma$, for some choice of $c_\delta \in \{\pi(\eta)^0_\delta, B_\Gamma\}$ for all $\delta \in \{-1, 0, 1\}$. Denote thus $P = \eta^{[M,2M-1]}_{[0,M-1]} \in C$.

Consider the cells η^{M-1}_δ for $\delta \in \{-1, 0, 1\}$ which lie directly below the base P^0_0 in η. Since P^0_0 is live, its Γ-state is obtained by applying Int_p, the local rule of Φ, to the Γ-states of these three cells. If η^{M-1}_{-1} is live, then we can prove by induction, using the rules of Ψ, that each η^t_{t-M} for $t \in [0, M-1]$ is a right moving cell, and they all have the same Γ-state, say $c \in \Gamma$. But this implies that $\eta^{[0,M-1]}_{[-M,-1]}$ is a macro-cell in state c, and then $\pi(\eta)^0_{-1} = c$. Analogously, we can show for all $\delta \in \{-1, 0, 1\}$ that if η^{M-1}_δ is live, then $\pi(\eta)^0_\delta$ equals its Γ-state. This finishes the proof. □

Lemma 6.3. *The sparse simulation of* Ψ *by* $\Phi = \mathrm{Int}^{\infty}_{G_s(n,p)}$ *is connecting.*

Proof. Let $\eta \in \tilde{\mathcal{T}}_\Psi$, and suppose $\pi(\eta)^t_i$ and $\pi(\eta)^{t-1}_{i+\delta}$ are adjacent for some $i, t \in \mathbb{Z}$ and $\delta \in \{-1, 0, 1\}$. We assume that $\delta = -1$, as the other cases are similar. We now have $\eta^{[tM,(t+1)M-1]}_{[iM,(i+1)M-1]}, \eta^{[(t-1)M,tM-1]}_{[(i-1)M,iM-1]} \in C$, and by the definition of C, η^{tM}_{iM} and $\eta^{(t-1)M}_{(i-1)M}$ are both right moving cells with counter value 0. Since $\pi(\eta)^t_i$ is (-1)-demanding, so is η^{tM}_{iM}, and thus η^{tM-1}_{iM-1} is a right moving cell with counter value $2n$. Right moving cells with nonzero counter value are all (-1)-demanding, and thus $\eta^{tM-\ell}_{iM-\ell}$ is a (-1)-demanding cell for all $\ell \in [0, 2n]$. This shows that the bases of the two macro-cells are directly connected. □

Next, we show that every live cell in a nondeterministic trajectory of Ψ originates from a macro-cell in the following sense.

Lemma 6.4. *Let* $\eta \in \tilde{\mathcal{T}}_\Psi$*, and let* η^t_i *be a live cell. Then* η^{t-s}_j *is the base of a macro-cell for some* $s \in [0, 2n]$ *and* $j \in [-n, n]$*, and is directly connected to* η^t_i*.*

Proof. First, if $\mathtt{Cnt}(\eta^t_i) = 0$, then η^t_i is necessarily the base of a macro-cell, and the claim holds. Otherwise, η^t_i is δ-demanding for some $\delta \in \{-1, 0, 1\}$, and the claim follows by induction. □

It can be easily proved by induction that if $\eta \in \tilde{\mathcal{T}}_\Psi$, and two cells η^t_i and η^s_j are weakly connected and have the same kind and counter value, then $|i - j|$ and $|t - s|$ are both divisible by M_n. The following two lemmas directly follow from this observation, and the former is the main reason for defining G_s in the first place.

Lemma 6.5. *Let $\eta \in \tilde{\mathcal{T}}_\Psi$, and let $(i,t) \in \mathbb{Z}^2$. Let $r \in \mathbb{N}$, and denote*

$$D(r) = \{s \in [0, r-1] \mid \eta_i^t \text{ is connected to } \eta_0^s\}.$$

Then $|D(r)| \leq \frac{3r}{n} + 3$.

Proof. Simply note that if two cells on the same column of η are connected and have the same kind, then they also have the same counter value, and use the above observation. □

Lemma 6.6. *Let $\eta \in \tilde{\mathcal{T}}_\Psi$, and let $(i,t),(j,s) \in \mathbb{Z}^2$ be such that $\eta_{[i,i+M_n-1]}^{[t,t+M_n-1]}$ and $\eta_{[j,j+M_n-1]}^{[s,s+M_n-1]}$ are macro-cells, and their bases are weakly connected. Then $|i-j|$ and $|t-s|$ are both divisible by M_n.*

Remark 6.1. In the following sections, up to and including the proof of the main theorem, we will not refer to the rules of any sparsified automaton, but only to the lemmas proved here, and the dimensions and base coordinate of the sparse simulations. Indeed, the exact implementation of the sparsification transformation is largely irrelevant.

7. Universal Programmatic Simulation

In this section, we construct a cellular automaton that simulates another automaton in a specific but highly modular way. The details of the construction are mainly borrowed from [7]. However, there is an important conceptual difference between the error-correcting, self-organizing simulator of Gács, and the following construction. Namely, whereas the Gács automaton is designed to detect and fix small errors in the simulation, our universal simulator is constructed so that every error will propagate and wipe out as much of the simulation structure as possible, replacing it by an area of blank cells. For this reason, it is also much simpler.

7.1. The Universal Simulator Automaton

We are now ready to present the construction for simulating a single arbitrary cellular automaton. For that, let p be a valid program that encodes the CA Int_p^∞ on the alphabet Γ. We construct a cellular automaton $\Psi : \Sigma^{\mathbb{Z}} \to \Sigma^{\mathbb{Z}}$ that simulates Int_p^∞ by presenting a valid program that defines Ψ. As with the sparsification transformation, there are many essentially similar ways of achieving universal programmatic simulation, and many details in our construction of Ψ are somewhat arbitrary.

The parameters and fields of Ψ are defined in Algorithm 3, and its main body is shown in Algorithm 4. The first field, `Live`, is called the *live flag*, and it has the same purpose as in the sparsification transformation. The procedure Die is also analogous to its earlier counterpart, making the current cell assume the state B_Σ.

The fields `Addr` and `Age`, respectively called *address* and *age*, are used to synchronize the behavior of distinct cells. Their maximum values are given by

Algorithm 3 The parameters and fields of Ψ, with comments indicating their purpose.

num param $\texttt{CSize} = Q$	▷ Size of colonies, width of macro-cells
num param $\texttt{WPeriod} = U$	▷ Length of work period, height of macro-cells
string param $\texttt{SimProg} = p$	▷ The program to be simulated
bool field `Live`	▷ Marks a cell that is not blank
num field $\texttt{Addr} \leq \texttt{CSize} - 1$	▷ Position of a cell in its colony
num field $\texttt{Age} \leq \texttt{WPeriod} - 1$	▷ Synchronizes the cells of a colony
bool field `Sim`	▷ Stores one bit of the state of Int_p^∞
enum field $\texttt{Work} \in (Q_\mathrm{A} \cup \{\#\}) \times \Sigma_\mathrm{A}$	▷ Contains the agent and/or its data
bool field `Prog`	▷ Stores one bit of `SimProg` for the agent
bool field `LMail`	▷ Retrieves the state of the left neighbor
bool field `RMail`	▷ Retrieves the state of the right neighbor
bool field `Out`	▷ Stores one bit of the output of the agent
bool field `LBord`	▷ Marks a left border cell
bool field `RBord`	▷ Marks a right border cell
bool field `Doomed`	▷ Marks a cell that dies at the end of the work period

the parameters Q, called the *colony size*, and U, the *work period*, respectively. The parameter Q is assumed to be divisible by 4. The age field is analogous to the `Cnt` field of the sparsification, while the address field serves the analogous purpose in the horizontal direction. The procedure VALID, shown in Algorithm 5, is used to determine whether the local structure of the simulation is correct. In particular, for cells with $\texttt{LBord} = \texttt{RBord} = \texttt{False}$, it returns `True` only if $\texttt{Age}^- = \texttt{Age} = \texttt{Age}^+$, $\texttt{Addr}^- + 1 \equiv \texttt{Addr} \equiv \texttt{Addr}^+ - 1 \bmod Q$ and $\texttt{Live}^- = \texttt{Live} = \texttt{Live}^+ = \texttt{True}$. Thus, in 'normal' conditions, the blank state spreads through live cells, though the fields `LBord` and `RBord` can change this behavior. Also, the address $\texttt{Addr}(x_i)$ of a cell $i \in \mathbb{Z}$ is normally $\texttt{Addr}(x_{i+1})$ decremented by one modulo Q, and adjacent cells have the same `Age` value. Following [7], a subword $w = x_{[i,i+Q-1]}$ of length Q is called a *colony*, if $\texttt{Addr}(w_j) = j$ for all $j \in [0, Q-1]$. The main idea of the construction is that each colony corresponds to a cell of the simulated automaton Int_p^∞, and explicitly computes a new state for itself every U steps, timed by the age field. The numbers Q and U are thus the dimensions of the simulation.

The next six fields (from `Sim` to `Out`) are used to programmatically simulate the automaton Int_p^∞. First, the Boolean field `Sim`, called the *simulation bit*, is used to store the simulated Γ-state of a colony $w \in \Sigma^Q$ as the bitstring $\texttt{Sim}(w_0)\texttt{Sim}(w_1)\cdots\texttt{Sim}(w_{\|\Gamma\|-1}) \in \{0,1\}^{\|\Gamma\|} = \Gamma$.

The work period is divided into two phases: the *retrieval phase* and the *computation phase*. The retrieval phase covers the first Q steps of the work period, and is governed by the rules of the `LMail` and `RMail` fields, called the *left and right mail fields*, respectively. This phase is entered in line 4.10, and it consists of moving the contents of the right and left mail fields Q steps to the

Algorithm 4 The program body of Ψ, and the procedure DIE.

```
1: procedure DIE
2:     Live ← False
3:     Addr, Age ← 0
4:     Sim, Prog, LMail, RMail, Out ← 0
5:     Work ← (#, B_A)
6:     LBord, RBord, Doomed ← False
7: if Live then
8:     if ¬VALID then DIE                          ▷ Destroy locally incorrect structure
9:     else
10:        if 0 ≤ Age < CSize then
11:            LBord, RBord ← False                ▷ Remove unnecessary border flags
12:            LMail ← LMail⁻                      ▷ Transfer information between colonies
13:            RMail ← RMail⁺
14:        if Age = CSize then
15:            Doomed ← False                      ▷ Remove existing doom flags
16:            if LBord ∧ Live⁻ then               ▷ Remove unnecessary border flags
17:                LBord ← False
18:            else if RBord ∧ Live⁺ then
19:                RBord ← False
20:            if Addr = 0 then                    ▷ Initialize computation
21:                Work ← (q_init, B_A)            ▷ Initial state and blank tape letter
22:            else
23:                Work ← (#, B_A)                 ▷ No head and blank tape letter
24:            Prog ← SimProg_Addr                 ▷ Returns 0 if Addr ≥ |SimProg|
25:        if CSize < Age < WPeriod − 1 then       ▷ Compute new state
26:            COMPUTE
27:        if Age = WPeriod − 1 then
28:            if Addr = 0 ∧ Doomed⁻ then          ▷ Prepare against dying neighbors
29:                LBord ← True
30:            else if Addr = CSize − 1 ∧ Doomed⁺ then
31:                RBord ← True
32:            if Doomed then DIE                  ▷ Kill a doomed colony
33:            else
34:                Sim, RMail, LMail ← Out         ▷ Assume new state
35:        Age ← Age + 1 mod WPeriod
36: else BIRTH                                     ▷ Become a live cell if necessary
```

Algorithm 5 The rules of local validity of a live cell in the universal simulation.

```
1:  procedure VALID
2:      if (Live⁻ = LBord) ∧ ¬(LBord ∧ RBord⁻ ∧ CSize/Age ∈ {1, 2}) then
3:          return False
4:      else if (Live⁺ = RBord) ∧ ¬(RBord ∧ LBord⁺ ∧ CSize/Age ∈ {1, 2}) then
5:          return False
6:      else if LBord⁺ ∨ RBord⁻ then
7:          return False
8:      else if Live⁻ ∧ (Addr⁻ + 1 ≢ Addr mod CSize ∨ Age⁻ ≠ Age) then
9:          return False
10:     else if Live⁺ ∧ (Addr ≢ Addr⁺ − 1 mod CSize ∨ Age ≠ Age⁺) then
11:         return False
12:     else if LBord ∧ (Age ≥ CSize) ∧ (Addr ≠ 0) then
13:         return False
14:     else if RBord ∧ (Age ≥ CSize) ∧ (Addr ≠ CSize − 1) then
15:         return False
16:     else
17:         return True
```

left and right, respectively. In a correct simulation, both mail fields are equal to the simulation bit in all cells at the beginning of the work period, and at the end of the retrieval phase, the left (right) mail fields of a colony then contain exactly the simulation bits of its left (right, respectively) neighbor.

In the computation phase, a Turing machine head, called the *agent*, is simulated on the cells of the colony, using the `Work` fields, called the *workspace fields*. These fields contain elements of $(Q_\mathrm{A} \cup \{\#\}) \times \Sigma_\mathrm{A}$, where Q_A is the state set of the agent, and $\Sigma_\mathrm{A} \times \{0,1\}^4$ is its tape alphabet. The component $\{0,1\}^4$ consists of the `LMail`, `RMail`, `Sim` and `Prog` fields, which the agent can access (but not modify) during its computation. At the beginning of the computation phase, the program p is stored in the `Prog` bits (the *program bits*) of the first $|p|$ cells of every colony $w \in \Sigma^Q$, so that $\texttt{Prog}(w_0)\texttt{Prog}(w_1)\cdots\texttt{Prog}(w_{|p|-1}) = p$ (line 4.24, where $p_i = 0$ if $i \geq |p|$). At age Q, the workspace fields of every cell of the colony except the leftmost one are cleared, and the agent is placed on the leftmost cell in its initial state. The procedure COMPUTE performs one step of computation of the agent. During the phase, the agent simulates the interpreter Int on the contents of the `LMail`, `Sim`, `RMail` and `Prog` fields of the colony, considering them as binary read-only tapes, and using the Σ_A-components of the workspace fields as a read-and-write tape. After at most $P^T_{\mathrm{Int}}(|p|, \|p\|)$ steps, it writes the result in the `Out` fields (the *output bits*) of the colony. If there is not enough room to perform the computation, the procedure DIE is called.

The three remaining fields, `Doomed`, `LBord` and `RBord` are called the *doom flag*, *left border flag* and *right border flag*, respectively, and their purpose is to make Ψ simulate the blank state of Int^∞_p using blank states, as in the sparsification transformation. The 'default value' of these fields is `False`, and a true value indicates some special circumstance.

In order to simulate blank states using blank states, some colonies have to be killed, and we choose to destroy those that are about to enter the state B_Γ. After completing the simulation of Int_p^∞ in the computation phase, the agent checks whether the new state of the colony would be B_Γ, and if so, it sets the doom flag of every cell in the colony to `True`. Otherwise, the flags remain `False`. On line 4.32, it is stated that a cell with age $U-1$ and doom flag `True` will immediately die. Thus every colony trying to switch to the blank state will be wiped out at the end of its work period, being replaced by a segment B_Σ^Q of blank cells.

Now, we also need a way to reclaim the blank segments, and this is where the border flags come in. Cells with age less than Q and left (right) border flag `True` are called *left (right, respectively) creative*. The purpose of creative cells is to allow the simulation to create a new colony in the place of a destroyed one. In the retrieval phase, a colony whose right neighbor has been replaced with blank cells will 'push' its simulation bits into the blank segment, rebuilding the neighbor (who still simulates the blank state) from the left one cell at a time. At any given time, the rightmost cell created in the new colony is right creative, but the others are not. Thus the creative status slides to the right at speed 1, or in other words, the only right creative cells in the unfinished colony with address $a \in [0, Q-1]$ has age $a+1$. To achieve this, a right creative cell whose right neighbor is blank is considered valid (and is not killed on line 4.8), provided that it has a consistent left neighbor and the above relation holds for its address and age. Also, according to procedure VALID, a creative cell must have a blank neighbor in order to be structurally valid and survive, unless its age is exactly $\frac{1}{2}Q$ (it is necessary to allow this special case, since a colony may be created from both sides simultaneously). The analogous rules hold for left creative cells.

This process of constructing a new colony is called *colony creation*, and it is governed by the procedure BIRTH (Algorithm 6), which is only called on blank cells, and line 4.11 of the main program. By the procedure BIRTH, if a blank cell has a left (right) neighbor which is right (left) creative (see lines 6.3 and 6.9), then it will become live, in a state that is consistent with that of the neighbor, and with `Sim`-bit and left (right, respectively) mail field 0. Furthermore, the new cell will get a `True` border flag itself. According to line 4.11, a creative cell always ceases to be creative on the next step, which causes only the bordermost cell of the creation process to be creative at any given time. These rules cause a new colony to be created during the retrieval phase next to an existing colony surrounded by blank cells on either side. The simulated state of the new colony is B_Γ, and its mail fields contain the simulation bits of the neighboring colonies that created it. To initialize the creation process, when a colony of doomed cells is about to be replaced by B_Σ^Q, the border cells of the neighboring colonies become creative to protect them from the resulting blank cells (lines 4.29 and 4.31).

If several neighboring colonies have been killed at the same time, a newly created colony may end up next to another blank segment, and needs to protect itself from it. This is also achieved with the border flags. Cells with address 0 ($Q-1$), age at least Q and left (right) border flag `True` are called *left (right,*

respectively) blocking. As with creative cells, a left (right) blocking cell whose left (right) neighbor is in state B_Σ is considered valid by the procedure VALID, provided that its address and age are consistent with those of its right (left, respectively) neighbor. Up to one exception, these are the only cells with age at least Q and border flag `True` that are considered valid. The one exception is given by a left (right) blocking cell of age Q surrounded by live cells with consistent addresses and ages, the leftmost (rightmost) of which is also right (left, respectively) blocking. Such cells will also not die, but simply become non-blocking. This situation happens when two neighboring colonies are created simultaneously. At the end of the computation phase, blocking cells with age $U-1$ that are not doomed become creative, and start the creation process of yet other colonies.

Algorithm 6 The rules for the creation of live cells cells.

```
1:  procedure BIRTH                          ▷ Executed if and only if ¬Live holds
2:      DIE                                  ▷ Guarantees that dead cells are blank
3:      if LBord⁺ ∧ (Age⁺ = CSize − Addr⁺ mod CSize) then
4:          Live ← True
5:          Addr ← Addr⁺ − 1 mod CSize
6:          Age ← Age⁺ + 1
7:          RMail ← RMail⁺
8:          LBord ← True
9:      else if RBord⁻ ∧ (Age⁻ = Addr⁻ + 1 mod CSize) then
10:         Live ← True
11:         Addr ← Addr⁻ + 1 mod CSize
12:         Age ← Age⁻ + 1
13:         LMail ← LMail⁻
14:         RBord ← True
```

We now make some useful definitions and observations about the local rule of the universal simulator automaton. They can be verified by inspecting the program of Ψ.

Definition 7.1. Let P be a pattern over Σ, finite or infinite. Two cells P_i^t and P_j^s of P are *consistent*, if they are live and satisfy $\texttt{Addr}(P_i^t) - \texttt{Addr}(P_j^s) \equiv i - j \bmod Q$ and $\texttt{Age}(P_i^t) - \texttt{Age}(P_j^s) \equiv t - s \bmod U$. If $P \in \Sigma^{Q\times U}$, we say that the cell P_i^t is *consistent with* P, if it is live and satisfies $\texttt{Addr}(P_i^t) = i$ and $\texttt{Age}(P_t^i) = t$. If P is a subpattern of a larger pattern, we extend this terminology to cells outside it in the natural way.

A live cell with age in $[1, Q]$ and left (right) border flag `True` is called *left (right) co-creative*.

Another word for co-creative cells could have been 'created', since co-creative cells are created by creative cells during a correct simulation, but since the notion is technical, we chose a more distinguishable term. Note that a blocking cell with age Q is co-creative.

Note that in the following observations, we are dealing with a trajectory of the nondeterministic CA $\hat{\Psi}$, which may regard any cell in a neighborhood as $\mathcal{B}_\Sigma$, independently of the other cells and the other neighborhoods.

Observation 7.1. Let $\eta \in \hat{\mathcal{T}}_\Psi$, let $(i,t) \in \mathbb{Z}^2$, and denote $c = \eta_i^t$.

- If c is left creative, then $\texttt{Age}(c) = Q - \texttt{Addr}(c) \bmod Q$. If $d = \eta_{i-1}^{t+1}$ is also consistent with c, then either d is left co-creative, or $\texttt{Age}(c) = \frac{1}{2}Q$ and η_{i-1}^t is consistent with c and right co-creative.
- If c is left co-creative, then $\texttt{Age}(c) = Q - \texttt{Addr}(c)$, and η_{i+1}^{t-1} is consistent with c and left creative.

The symmetric claim, with the left and right notions switched and $Q - \texttt{Addr}(c)$ replaced by $\texttt{Addr}(c) + 1$, also holds.

Observation 7.2. Let $\eta \in \hat{\mathcal{T}}_\Psi$ and $P = \eta_{[0,Q-1]}^{[0,U-1]}$. Let $\delta \in \{-1, 0, 1\}$, and let $(i,t) \in [0, Q-1] \times [0, U-1]$ be such that exactly one of $c = \eta_{i+\delta}^{t+1}$ and η_i^t is consistent with P. Then at least one of the following conditions holds:

- c is blank;
- c is co-creative, and $t \in [0, Q-1]$;
- $\delta = -1$, c is right blocking and $i + \delta \equiv Q - 1 \bmod Q$;
- $\delta = -1$, $\eta_{i+\delta}^t$ is right co-creative, and $t \in [1, Q]$;
- $\delta = 1$, c is left blocking and $i + \delta \equiv 0 \bmod Q$; or
- $\delta = 1$, $\eta_{i+\delta}^t$ is left co-creative, and $t \in [1, Q]$.

Finally, we prove that the automaton Ψ actually simulates Int_p^∞ with the parameters Q and U, provided that they are sufficiently large. Recall that $\|p\|$ is defined as $\|\Gamma\|$.

Lemma 7.1. *Define $C \subset \Sigma^{Q \times U}$ as the set of rectangular patterns $P \in \Sigma^{Q \times U}$ whose $2Q$'th row from the top satisfies $\texttt{Live}(P_i^{U-2Q}) = \texttt{True}$, $\texttt{Age}(P_i^{U-2Q}) = U - 2Q$ and $\texttt{Addr}(P_i^{U-2Q}) = i$ for all $i \in [0, Q-1]$. For all $P \in C$, define $\pi(P) = \texttt{Sim}(P_0^{U-2Q})\texttt{Sim}(P_1^{U-2Q}) \cdots \texttt{Sim}(P_{\|p\|-1}^{U-2Q}) \in \Gamma$, the word formed by the $\texttt{Sim}$-bits of the first $\|p\|$ cells of said row. For $P \in \Sigma^{Q \times U} \setminus C$, define $\pi(P) = B_\Gamma$. If the values Q and U satisfy*

$$Q \geq P_{\mathrm{Int}}^S(|p|, \|p\|) \tag{2}$$

and

$$U \geq 6Q + P_{\mathrm{Int}}^T(|p|, \|p\|), \tag{3}$$

then $(Q, U, (0, U-2Q), C, \pi)$ is a simulation of Int_p^∞ by Ψ.

Proof. Denote $\Phi = \mathrm{Int}_p^\infty$. The condition $\pi(\Sigma^{Q\times U} \setminus C) = B_\Gamma$ holds by definition, so we only need to show $\mathcal{T}_\Phi^+ \subset \pi(\mathcal{T}_\Psi^+)$. For that, let $\eta \in \mathcal{T}_\Phi^+$ be arbitrary. Define the one-dimensional configuration $x \in \Sigma^{\mathbb{Z}}$ as follows. For each $i \in \mathbb{Z}$ and $j \in [0, Q-1]$, define the cell $s = x_{iQ+j} \in \Sigma$ by

$$\begin{array}{rclrclrcl}
\texttt{Live}(s) & = & \texttt{True}, & \texttt{Out}(s) & = & 0, & \texttt{Sim}(s) & = & (\eta_i^0)_j, \\
\texttt{Addr}(s) & = & j, & \texttt{Work}(s) & = & (\#, \Sigma_\mathrm{A}), & \texttt{LMail}(s) & = & \texttt{Sim}(s), \\
\texttt{Age}(s) & = & 0, & \texttt{Prog}(s) & = & 0, & \texttt{LMail}(s) & = & \texttt{Sim}(s), \\
\texttt{LBord}(s) & = & \texttt{False}, & \texttt{RBord}(s) & = & \texttt{False}, & \texttt{Doomed}(s) & = & \texttt{False}.
\end{array}$$

Now, each segment $x_{[iQ,(i+1)Q-1]} \in \Sigma^Q$ is a colony with age 0 and Γ-state η_i^0. Since the mail fields of these colonies have the same contents as their simulation fields, the information retrieved by their neighbors is correct, and after U steps, every colony computes itself a new state. The inequality (2) guarantees that the agents have enough space to perform the computations, while (3) guarantees the time. There are also $6Q$ extra steps coming from the retrieval phase, the marking of doomed cells, the fact that the $2Q$'th row from the top must consist of live cells, and some extra steps needed in the proofs of Section 7.2 and Section 9. If we let $\xi \in \mathcal{T}_\Psi^+$ be the trajectory whose bottom row is x, this shows that $\pi(\xi)^0 = \eta^0$, that is, the bottom rows of $\pi(\xi)$ and η coincide.

Now, we show by induction that $\pi(\xi)^t = \eta^t$ holds for all $t \in \mathbb{N}$. More explicitly, we show that for all $i \in \mathbb{Z}$, the segment $\xi^{(t+1)U-1}_{[iQ,(i+1)Q-1]} \in \Sigma^Q$ is either a colony with age $U-1$, containing the state η_i^t in its $\texttt{Sim}$ fields and η_i^{t+1} in its $\texttt{Out}$ fields, or the blank segment $\mathcal{B}_\Sigma^Q$, which cannot border a colony in a non-blank state. In the first case, if $\eta_i^{t+1} = B_\Gamma$, then the cells of the colony are doomed, and it will be destroyed in the next step. The latter case can only happen when $\eta_i^t = B_\Gamma$. If a doomed colony or a blank segment lies next to a colony that is not doomed, a new colony will be created in its place in the next Q steps that form the retrieval phase. During this phase, the colonies will also retrieve the state data of their neighbors. During the computation phase, the colonies calculate new states for themselves, and the blocking cells keep the blank areas from spreading over the colonies. From the induction hypothesis, and the fact that B_Γ is quiescent for Φ, it follows that these states form the row η^{t+1}. □

If the bounds in Equations (2) and (3) hold, we denote $\Psi = \mathrm{Univ}_p^{Q,U}$ (or just Univ_p, if Q and U are clear from the context or irrelevant), and call Ψ a *universal simulator automaton (with input p)*. The choice for the $2Q$'th row from the top, as opposed to the top row, may seem arbitrary, but it is actually necessary in some of the results of Section 7.2. In all applications of universal simulators, the dimensions Q and U are assumed to be large (the bound $Q \geq 10$ should be sufficient). Since P_Int^T grows faster than P_Int^S, the parameter U will usually be orders of magnitude larger than Q.

In Figure 3, we have illustrated the simulation of the three-neighbor binary XOR automaton by a universal simulator, in a schematic form.

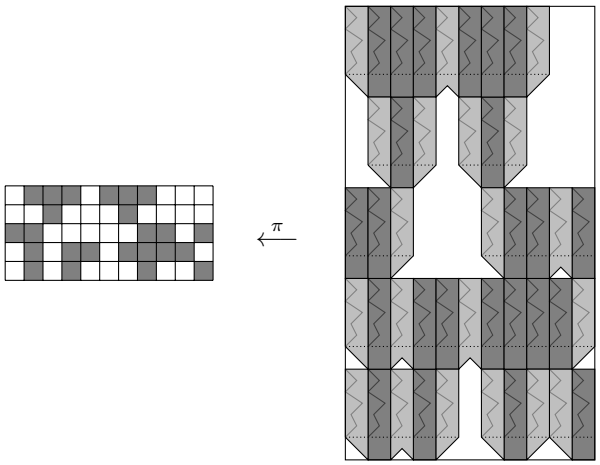

Figure 3: Simulating the three-neighbor XOR automaton. The dotted lines represent transitions between retrieval and computation phases, and the jagged lines represent the agents. Dark macro-cells have state 1, and light ones have 0.

7.2. Properties of Universal Simulation

Next, we show that simulation by a universal simulator automaton is very well-behaved. For this section, let $\Psi = \mathrm{Univ}_p^{Q,U}$ be a universal simulator CA for the automaton Int_p^∞, using the simulation $(Q, U, (0, U-2Q), C, \pi)$. Be begin with the following rather technical lemma that restricts the structure of those macro-cells that actually appear in nondeterministic trajectories of Ψ.

Lemma 7.2. *Let $\eta \in \hat{\mathcal{T}}_\Psi^+$ be such that $P = \eta_{[0,Q-1]}^{[U,2U-1]} \in C$. Then for all $t \in [Q, U-2Q]$, the row P^t is a colony with age t and simulated state $\pi(P) \in \Gamma$. Also, exactly one of the following conditions holds.*

1. *$\pi(P) \neq B_\Gamma$, and for all $t \in [0, Q-1]$, P^t is a colony with age t and simulated state $\pi(P)$, and P contains no co-creative cells. Also, the $Q \times U$ rectangle below P satisfies $\eta_{[0,Q-1]}^{[0,U-1]} \in C$.*
2. *$\pi(P) = B_\Gamma$, and for all $t \in [0, Q-1]$ and $i \in [t-Q, t-1]$, the cell η_i^{U+t} is consistent with P, and η_{t-1}^{U+t} is right creative.*
3. *$\pi(P) = B_\Gamma$, and for all $t \in [0, Q-1]$ and $i \in [Q-t, 2Q-t-1]$, the cell η_i^{U+t} is consistent with P, and η_{Q-t}^{U+t} is left creative.*
4. *$\pi(P) = B_\Gamma$, for all $t \in [0, Q-1]$ and $i \in [t-Q, t-1] \cup [Q-t, 2Q-t-1]$, the cell η_i^{U+t} is consistent with P. Also, the cell η_{t-1}^{U+t} (η_{Q-t}^{U+t}) is right (left, respectively) creative for all $t \in [0, \frac{1}{2}Q]$.*

The lemma intuitively states that a macro-cell consists mostly of live cells that are consistent with it, and that there are four possibilities for the shape of its Q lowest rows. The four cases correspond to P not being created, being created from the left, created from the right, and created from both directions, respectively. Figure 3 contains examples of all the cases.

Proof. The first claim is an easy induction argument, since P^{U-2Q} is of said form by the definition of C, and if P^{t-1} is not of that form, then neither is P^t, by Observations 7.1 and 7.2. Namely, if a cell P_i^{t-1} is inconsistent with P, then P_i^t must be co-creative, which is impossible since $t > Q$.

Consider then the border cells of P, and suppose that P_0^t and P_{Q-1}^s are left and right co-creative, respectively, and consistent with P, for some $t, s \in [1, Q]$. Then necessarily $t = s = Q$ by Observation 7.1. We can show by induction that P_{t-1}^t is right co-creative and P_{Q-t}^t is left co-creative for all $t \in [\frac{1}{2}Q, Q-1]$, but then the row $P^{\frac{1}{2}Q}$ has no preimage under $\hat{\Psi}$, a contradiction. Thus both borders cannot contain co-creative consistent cells.

Next, suppose that the right border cell P_{Q-1}^Q is right co-creative and consistent with P. Now we can inductively show, using the two observations and the rules of Ψ, that for all $t \in [0, Q-1]$ and $i \in [t-Q, t-1]$, the cell P_i^t is consistent with P (this is an induction first on i, then on t). Furthermore, Observation 7.1 shows that P_{t-1}^t is right creative and thus satisfies $\texttt{Sim}(P_t^{t+1}) = 0$. This in turn implies that $\pi(P) = B_\Gamma$, since $\texttt{Sim}$-bits do not change during the work period, and we are in the second case above. Symmetrically, if the left cell P_0^Q is left co-creative and consistent, we are in the third case.

Suppose finally that neither border cell is both consistent and co-creative. Then the two observations show that P_0^t and P_{Q-1}^t are consistent for all $t \in [1, Q-1]$. Furthermore, each row P^t with $t \in [\frac{1}{2}Q + 1, Q-1]$ consists of cells consistent with P, and none of them are creative.

Observation 7.1 gives two possibilities for the row $w = P^{\frac{1}{2}Q}$: either $w_{\frac{1}{2}Q-1}$ is right co-creative and $w_{\frac{1}{2}Q}$ is left co-creative, in which case we can prove as above that the fourth case holds, or w does not contain co-creative cells. In the latter case, the induction can continue, and P^t is a colony with age t for all $t \in [0, \frac{1}{2}Q]$. It remains to show that in this case we have $\pi(P) \neq B_\Sigma$ and $R = \eta_{[0,Q-1]}^{[0,U-1]} \in C$, and for that, note that the top row of R is a colony with age $U-1$ that cannot be doomed. As in the proof of the first claim of this lemma, we can show that R^t is a colony with age t for all $t \in [Q, U-1]$, and thus $R \in C$ and the agent of R correctly computes $\pi(P)$ from the different fields of R. Now, if we had $\pi(P) = B_\Sigma$, then the agent of R would have marked the cells of R^{U-1} with the doom flag, a contradiction. □

As an immediate corollary, we obtain that two distinct macro-cells cannot overlap very much. This corollary is later used to prove that in a trajectory our final sparse automaton, there cannot exist too many simultaneous simulation structures that are not consistent with each other.

Corollary 7.1. *Let $\eta \in \hat{\mathcal{T}}_\Psi$, and suppose we have $\eta_{[i,i+Q-1]}^{[t,t+U-1]} \in C$ and $\eta_{[j,j+Q-1]}^{[s,s+U-1]} \in C$ with $|i-j| < Q$ and $|t-s| < U - 3Q$. Then $(i,t) = (j,s)$.*

Using Lemma 7.2 and the two observations from Section 7.1, we can prove the strong rigidity and strong connectivity of the universal simulation.

Lemma 7.3. *The simulation of $\Phi = \mathrm{Int}_p^\infty$ by $\Psi = \mathrm{Univ}_p^{Q,U}$ is strongly rigid.*

Proof. Let $\eta \in \hat{\mathcal{T}}_\Psi$ be a trajectory such that $P = \eta_{[0,Q-1]}^{[U,2U-1]} \in C$. It is enough to prove that $\mathrm{Int}_p(\pi(\eta)_{-1}^0, \pi(\eta)_0^0, \pi(\eta)_1^0) = \pi(P)$ holds if the simulated state of P is not blank, or $\pi(P) \neq B_\Gamma$. By the first case of Lemma 7.2, we have that $R = \eta_{[0,Q-1]}^{[0,U-1]} \in C$ and that the agent of R correctly computes $\pi(P)$ from the inputs it receives. Thus it suffices to prove that the inputs are correct, that is,

$$\texttt{LMail}(R_i^Q) = (\pi(\eta)_{-1}^0)_i,\ \texttt{Sim}(R_i^Q) = (\pi(\eta)_0^0)_i,\ \texttt{RMail}(R_i^Q) = (\pi(\eta)_1^0)_i \qquad (4)$$

for all $i \in [0, Q-1]$. Since $\pi(\eta)_0^0 = \pi(R)$, the second condition holds for all i, and since the remaining cases are symmetric, it suffices to prove the first one.

Denote by $T = \eta_{[-Q,-1]}^{[0,U-1]}$ the $Q \times U$ rectangular pattern to the left of R. We split the proof into three cases depending on the simulated state of T and the structure of R. See Figure 4 for illustrations of the cases, where the dotted lines represent the cells R_i^Q for $i \in [0, Q-1]$.

Case 1: $\pi(T) \neq B_\Gamma$. In this case, we necessarily have $T \in C$, and the first case of Lemma 7.2 holds for T. In particular, the bottom row of T contains the correct simulated state on its `Sim`-bits and thus also on its `LMail` fields, or $\texttt{LMail}(T_i^0) = (\pi(\eta)_{-1}^0)_i$ for all $i \in [0, Q-1]$. Also, the cell $c = T_{Q-1}^Q$ on the right border of T (the small square in Figure 4) is not co-creative. This implies that R_0^Q, the right neighbor of c, is not co-creative either, as otherwise the procedure VALID would return `False` for c (on line 5.6), and the cell T_{Q-1}^{Q+1} above c would be blank, contradicting Lemma 7.2. Thus R is not created from the right, and Lemma 7.2 implies that the horizontal segment $\eta_{[t-Q,t-1]}^t$ (between the dashed lines in Figure 4) consists of cells which are consistent with T and R for all $t \in [0, Q]$. By induction on t, we can prove that $\texttt{LMail}(T_{Q-t+i}^t) = (\pi(\eta)_{-1}^0)_i$ for all $i \in [0, Q-1]$ and $t \in [0, Q]$. Namely, each cell η_i^t simply copies its `LMail` field from η_{i-1}^{t-1}. We have $\texttt{LMail}(R_i^Q) = (\pi(\eta)_{-1}^0)_i$ in the final case $t = Q$, which is exactly the first claim of (4).

Case 2: $\pi(T) = B_\Gamma$ and R is created from the right. In other words, R satisfies case 3 of Lemma 7.2. This implies that for all $t \in [1, Q]$, the cell R_{Q-t}^t is left creative (represented by the solid diagonal line in Figure 4) and R_{Q-t}^{t-1} is blank. We prove by induction on $t + i$ that for all $t \in [1, Q]$ and $i \in [Q-t, Q-1]$, we have $\texttt{LMail}(R_i^t) = 0$. Namely, this holds for the left creative cells and their right neighbors (the base cases $i = Q - t$ and $i = Q - t + 1$), which copy their `LMail` fields from blank cells (or cells that the local rule of $\hat{\Psi}$ regards as blank). The inductive step works as in Case 1. When $t = Q$, we again have the first claim of (4).

Case 3: $\pi(T) = B_\Gamma$ and R is not created from the right. In a correct simulation, this case occurs only when the $Q \times U$ rectangle under T is a doomed macro-cell, and the $Q \times U$ rectangle under R is a macro-cell which is not doomed. Now, since R is not created from the right, the left border cell R_0^Q is not left blocking. By induction, none of the cells R_0^t for $t \in [Q, U-1]$ is left blocking. We prove

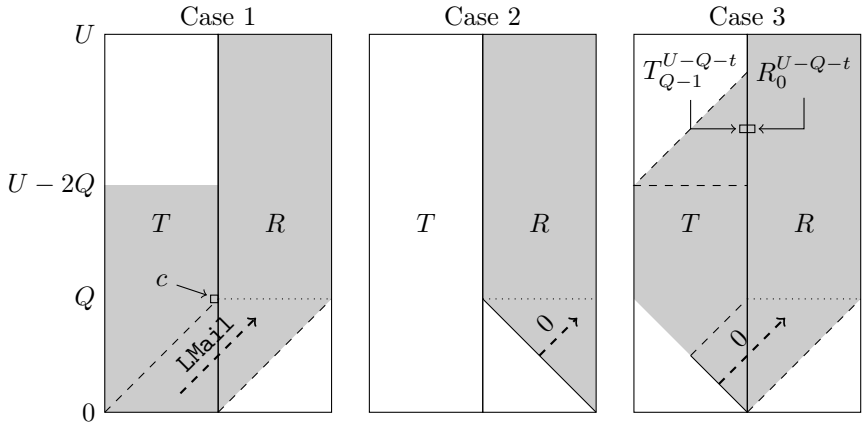

Figure 4: An illustration of the proof of Lemma 7.3, not drawn to scale. The gray areas represent cells that we have proved to be live.

by induction on i that all of the cells T_i^{U-Q-t} for $t \in [1, Q]$ and $i \in [Q-t, Q-1]$ (the upper dashed triangle in Figure 4) are consistent with T. Consider first a cell $T_{Q-1}^{U-Q-t} = \eta_{-1}^{U-Q-t}$ on the right border of T, which is the left neighbor of R_0^{U-Q-t} on the left border of R. These cells are represented by the small squares in Figure 4. Since the cell $R_0^{U-Q-t+1}$ is consistent with T but not left blocking, Observation 7.2 applied to $\delta = 1$ implies that T_{Q-1}^{U-Q-t} is also consistent with T. The inductive step is a similar application of Observation 7.2. In the final case $t = Q$, we see that every cell in the row T^{U-2Q} (the horizontal dashed line) is consistent with T, so that $T \in C$ by the definition of C. Now, T cannot satisfy the first case of Lemma 7.2 since $\pi(T) = B_\Gamma$, and it cannot be created from the left for the same reasons as R could not be created from the right in Case 1. This in turn implies that the cell $T_{Q-t}^t = \eta_{-t}^t$ is left co-creative for all $t \in [1, \frac{1}{2}Q]$ (the solid diagonal line in the figure). Then we can prove similarly to Case 2 that $\texttt{LMail}(\eta_i^t) = 0$ for all $t \in [1, Q]$ and $i \in [\max(-t, -Q+t), t-1]$ (the area between the lower dashed lines), which again implies the first claim of (4) when $t = Q$. □

Lemma 7.4. *The simulation of* $\Phi = \mathrm{Int}_p^\infty$ *by* $\Psi = \mathrm{Univ}_p^{Q,U}$ *is strongly connecting.*

Proof. Let $\eta \in \hat{\mathcal{T}}_\Psi$ be such that $P = \eta_{[0,Q-1]}^{[U,2U-1]} \in C$ and $\pi(P)$ is δ-demanding for some $\delta \in \{-1, 0, 1\}$. In particular, we have $\pi(P) \neq B_\Sigma$, and Lemma 7.2 then implies that $R = \eta_{[0,Q-1]}^{[0,U-1]} \in C$. Then every cell of the rectangle $\eta_{[0,Q-1]}^{[Q,2U-2Q]}$ is consistent with P. Also, the cells of the vertical column $V = \eta_0^{[U-2Q,2U-2Q]}$ on the left border of P and R are all 0-demanding, so we have a directed path from the base of P to that of R, which proves the claim when $\delta = 0$.

Suppose then that $\delta = -1$. Since Ψ is strongly rigid by Lemma 7.3, we have $T = \eta_{[0,Q-1]}^{[0,U-1]} \in C$ and $\pi(T) \neq B_\Sigma$. In particular, the base cell T_0^{U-2Q} is consistent with T. As in the proof of Lemma 7.3, we can also prove that the left border of R does not contain left blocking cells. Using Observation 7.2, we can again prove by induction on t that every cell T_{Q-t}^{U-Q-t} for $t \in [1, Q]$ is consistent with T. Furthermore, the cells are (-1)-demanding, and the cell T_0^{U-2Q} of the case $t = Q$ is the base of T. Combined with the aforementioned column V, we have obtained a directed path from the base of P to that of T, which proves the claim when $\delta = -1$. The case $\delta = 1$ is even simpler. □

Finally, we prove the analogue of Lemma 6.4 from Section 6.2 for the universal simulation.

Lemma 7.5. *Let $\eta \in \hat{\mathcal{T}}_\Psi$, and let $c = \eta_i^t$ be a live cell for some $(i, t) \in \mathbb{Z}^2$. Then for some $s \in [t - 2U, t - U + 1]$ and $j \in [i - 3Q + 1, i + 2Q - 1]$, the rectangle $P = \eta_{[j,j+Q-1]}^{[s,s+U-1]}$ is a macro-cell in a non-blank state, and c is directly connected to the base of P, and thus consistent with it.*

Proof. Suppose first that $\texttt{Age}(c) = U - 1$. In this case, we claim that the result holds for $s = t - U + 1$ and some $j = j_\delta = i - \texttt{Addr}(c) + \delta Q$ for $\delta \in \{-1, 0, 1\}$. Namely, we can prove by induction that for all $r \in [0, 2Q - 1]$, the number of cells on the row $\eta_{[j_0,j_0+Q-1]}^{t-r}$ that are directly connected to (and thus consistent with) c is at least $\min(Q, r + 1)$. For $r = 2Q - 1$ this implies that the row $\eta_{[j_0,j_0+Q-1]}^{t-2Q+1}$ is a colony with age $U - 2Q$, so that $R = \eta_{[j_0,j_0+Q-1]}^{[t-U+1,t]} \in C$. If $\pi(R) \neq B_\Sigma$, we can choose $P = R$ and the claim holds.

Otherwise, R is created from some direction, and we assume that it is created from the left, the other case being symmetric. Since the left border of R does not contain blocking cells, we can prove that for all $r \in [0, Q - 1]$, the row $\eta_{[j_0-r,j_0+Q-1-r]}^{t-Q-r}$ consists of cells that are directly connected to c. In the final case $r = Q - 1$, we see that $\eta_{[j_{-1},j_{-1}+Q-1]}^{t-2Q+1}$ is also a colony with age $U - 2Q$. Thus $T = \eta_{[j_{-1},j_{-1}+Q-1]}^{[t-U+1,t]} \in C$, and it is easily seen that T cannot be created from either direction, implying that $\pi(T) \neq B_\Sigma$. Then we can choose $P = T$.

Next, suppose that $a = \texttt{Age}(\eta_i^t) \neq U - 1$. If $a \in \{0\} \cup [Q + 1, U - 2]$, then Observation 7.2 implies that η_i^{t-1} is consistent with η_i^t, and otherwise, Observation 7.1 implies that one of η_δ^{t-1} for $\delta \in \{-1, 0, 1\}$ is. Since the age of said consistent cell is $(a - 1) \bmod U$, a simple induction finishes the proof. □

8. Amplification

The main reason for constructing the universal simulator automaton is the following powerful result, which shows the existence of a wide class of amplifiers.

Theorem 8.1. *Let G be a polynomial CA transformation. Then there exist computable sequences $(Q_n, U_n)_{n\in\mathbb{N}}$ in $\mathbb{N}^2$ and $(p_n)_{n\in\mathbb{N}}$ in $\{0, 1\}^*$ such that each p_n is a program for the universal simulator $\mathrm{Univ}_{G(n+1,p_{n+1})}^{Q_n,U_n}$.*

In particular, if the transformation G is such that $\mathrm{Int}^\infty_{G(n,p)}$ always simulates Int^∞_p, then the sequence $(\mathrm{Int}^\infty_{G(n,p_n)})_{n\in\mathbb{N}}$ is an amplifier. We also note that the sequence $(Q_n, U_n)_{n\in\mathbb{N}}$ of dimensions could be chosen quite freely (here they are some polynomial functions of n), but we have no need for such fine-tuning in this article.

Proof. Let P_G be a polynomial in two variables such that $|G(n,p)| \leq P_G(n,|p|)$ and $\|G(n,p)\| \leq P_G(n,\|p\|)$ hold for all valid programs $p \in \{0,1\}^*$ and $n \in \mathbb{N}$. Since the programming language defined in Section 5 may be assumed to contain any particular polynomially computable function, we assume it contains the transformation G.

Algorithm 7 The modified parameters of p_n.

num param `Level` $= n$
num param `CSize` $= c_1 \cdot (\texttt{Level} + 1)^{c_2} - 1$
num param `WPeriod` $= c_3 \cdot (\texttt{Level} + 1)^{c_4} - 1$
string param `SimProg` $= G(\texttt{Level} + 1, \textsc{IncLevel}(\texttt{This}))$

We construct the programs p_n by slightly modifying the program of the universal simulator CA. In fact, it is enough to replace the parameter definitions of Algorithm 3 with Algorithm 7, and define one new procedure IncLevel. The modified program contains the new parameter `Level`, whose value is simply n, and the definitions of the other parameters are changed slightly. In particular, the simulated program is now calculated from p_n itself, using the keyword `This` (which refers to p_n as a bitstring) and the procedure IncLevel, which returns p_{n+1} by simply incrementing the `Level` parameter of its argument program by one. The numbers $c_1, c_2, c_3, c_4 \in \mathbb{N}$ are some constants that only depend on G. Of course, we define $Q_n = c_1(n+1)^{c_2}$ and $U_n = c_3(n+1)^{c_4}$ for all $n \in \mathbb{N}$.

We now claim that for some values of the constants c_i, the inequalities (2) and (3) hold with Q, U and p replaced by Q_n, U_n and $G(n+1, p_{n+1})$ for all $n \in \mathbb{N}$. For that, consider the program p_{n+1}. Since the numbers are encoded in binary, its length is asymptotically $|p_{n+1}| = \log_2 n + \sum_{i=1}^4 \log_2(c_i) + \mathcal{O}(1)$, since the fields, procedure definitions and main body of the program are all independent of n and the constants c_i. Also, for the length of the states of p_{n+1} we have $\|p_{n+1}\| = \log_2(c_1c_3) + (c_2 + c_4)\log_2 n + \mathcal{O}(1)$.

Denote $q_n = G(n+1, p_{n+1})$. We can now estimate the values $|q_n|$ and $\|q_n\|$ of the simulated automaton by

$$|q_n| = P_G(n, \log_2 n + \sum_{i=1}^4 \log_2(c_i)) + \mathcal{O}(1)$$

and

$$\|q_n\| = P_G(n, \log_2(c_1c_3) + (c_2 + c_4)\log_2 n) + \mathcal{O}(1).$$

This implies that the complexities $P^S_{\mathrm{Int}}(|q_n|, \|q_n\|)$ and $P^T_{\mathrm{Int}}(|q_n|, \|q_n\|)$ are polynomial in $\log_2 n$ and the constants c_i. It is clear that some values for the c_i

then exist that guarantee the inequalities $c_1(n+1)^{c_2} \geq P^S_{\mathrm{Int}}(|q_n|, \|q_n\|)$ and $c_3(n+1)^{c_4} \geq 6c_1(n+1)^{c_2} + P^T_{\mathrm{Int}}(|q_n|, \|q_n\|)$ for all $n \in \mathbb{N}$. But these are exactly the bounds in (2) and (3), and Lemma 7.1 finishes the proof. □

Theorem 8.1 reveals the motivation behind the definition of polynomial CA transformations. Namely, a consistent transformation G should be seen as a function that takes a cellular automaton Ψ, together with an integer parameter $n \in \mathbb{N}$, and endows it with a property P_n, so that $P_n(G(n, \Psi))$ holds. Now, the theorem implies that if $G(n, \Psi)$ is sufficiently similar to Ψ, in the sense of being able to simulate it, there exists a single CA (the bottom of the amplifier) that, at least in some sense, satisfies P_n for all $n \in \mathbb{N}$. We could say that this single CA *amplifies* the effects of G.

9. The Sparse Amplifier

9.1. Properties of the Sparse Amplifier

Let $N \subset \mathbb{N}$ be decidable in polynomial time. Theorem 8.1 implies the existence of an amplifier $(G^N_s(n, \Psi_n))_{n\in\mathbb{N}}$, where each Ψ_n is a universal simulator CA for $G^N_s(n+1, \Psi_{n+1})$. We call this the *sparse amplifier* with effective set N, and proceed to study its properties.

We establish some notation for the rest of this section. For convenience, we denote $\Phi_n = G^N_s(n, \Psi_n)$, and let Σ_n and Γ_n be the alphabets of Ψ_n and Φ_n, respectively. Let also $Q_n, U_n \in \mathbb{N}$ be given by Theorem 8.1, and let $S^u_n = (Q_n, U_n, C^u_n, \pi^u_n)$ be the simulation of Φ_{n+1} by Ψ_n. Denote $M_n = 2n+1$ if $n \in N$, and $M_n = 1$ otherwise, and let $S^s_n = (M_n, M_n, C^s_n, \pi^s_n)$ be the simulation of Ψ_n by Φ_n. Denote $Q'_n = Q_n M_n$ and $U'_n = U_n M_n$, and let $S'_n = (Q'_n, U'_n, C'_n, \pi'_n)$ be the simulation of Φ_{n+1} by Φ_n, that is, $S'_n = S^u_n \circ S^s_n$. We also denote $B_n = \prod_{i=0}^{n-1} Q'_n$ and $W_n = \prod_{i=0}^{n-1} U'_n$, and let $S_n = (B_n, W_n, C_n, \pi_n)$ be the composed simulation of Φ_n by Φ_0. In particular, S_0 is the identity simulation of Φ_0 by itself. Thus, the S^s_n are the sparse simulations, the S^u_n the universal simulations, the S'_n their compositions, and the S_n the n-fold compositions of the S'_n. The bottom of the amplifier, Φ_0, is our main object of interest, and its unique ergodicity will be proved in the next section. The simulation chain is drawn in Figure 5.

We now proceed to prove the main properties of the sparse amplifier. They mostly follow from the results of Section 6.2 and Section 7.2. In their proofs, we always assume that $N = \mathbb{N}$ unless otherwise noted, since the case $n \notin N$ is usually simpler than its converse and can be omitted.

Lemma 9.1. *Every simulation S'_n and S_n is rigid and connecting.*

Proof. This follows by induction from the rigidity (Lemma 6.2 and Lemma 7.3) and connectivity (Lemma 6.3 and Lemma 7.4) of the sparse and universal simulations, and Lemma 4.2. □

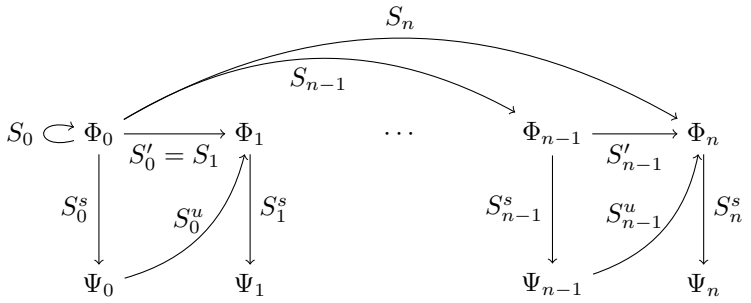

Figure 5: The chain of simulations induced by the sparse amplifier.

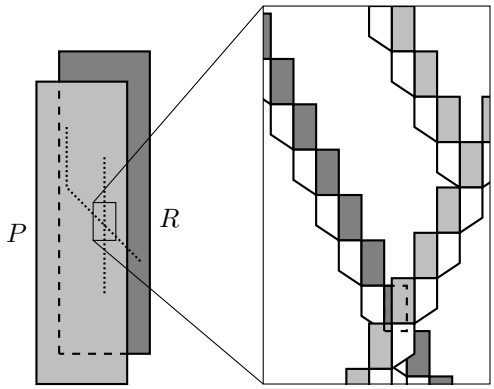

Figure 6: A visualization of Lemma 9.2. On the left, the overlapping macro-cells P and R, and the two chains of subcells. On the right, the non-blank macro-cells $P^{(s)}_{(j_s)}$ and $R^{(s)}_{(k_s)}$, and under them, the macro-cells $P^{(s-1)}_{(j_s)}$ and $R^{(s-1)}_{(k_s)}$, of the simulation S_{n-1}. Of these, some necessarily overlap, shown by the dashed border.

The following result is analogous to Corollary 7.1 from Section 7.2. It states that two distinct macro-cells of the simulation S_n cannot overlap by more than $\frac{1}{3}$ of their width and height, even in a nondeterministic trajectory of $\tilde{\Phi}_0$. The bound $\frac{1}{3}$ is not optimal, but suffices for our needs. Since this proof quickly becomes quite technical, we split it into several smaller claims. The reader should consult Figure 6 for a visualization of the proof.

Lemma 9.2. *Let $\eta \in \tilde{\mathcal{T}}_{\Phi_0}$ and $n \in \mathbb{N}$, and consider the simulation S_n of Φ_n by Φ_0. Suppose that we have $P = \eta^{[0,W_n-1]}_{[0,B_n-1]} \in C_n$ and $R = \eta^{[t,t+W_n-1]}_{[i,i+B_n-1]} \in C_n$ for some $i \in [0, \frac{2}{3}B_n]$ and $t \in [0, \frac{2}{3}W_n]$. Then $(i,t) = (0,0)$, so that $P = R$.*

Proof. By induction on n. For $n = 0$ the claim is trivial since the macro-cells of the simulation S_0 are just cells, so assume $n \geq 1$. Let $\xi = \pi_{n-1}(\eta)$, $\zeta = \pi^s_{n-1}(\xi)$ and $\theta = \pi^s_{n-2}(\pi_{n-2}(\eta))$ (if $n \geq 2$). Note that we have $\xi = \pi^u_{n-2}(\theta)$.

Since the simulations S_k and S_k^s are rigid and weakly rigid by Lemma 9.1 and Lemma 6.2, respectively, we have that $\xi \in \tilde{\mathcal{T}}_{\Phi_{n-1}}$, $\zeta \in \hat{\mathcal{T}}_{\Psi_{n-1}}$ and $\theta \in \hat{\mathcal{T}}_{\Psi_{n-2}}$. Next, we dissect the macro-cells P and R into smaller macro-cells of the lower-level simulation S_{n-1}. For all $(j,s) \in [0, Q'_{n-1} - 1] \times [0, U'_{n-1} - 1]$, denote $P^{(j)}_{(s)} = P^{[sW_{n-1},(s+1)W_{n-1}-1]}_{[jB_{n-1},(j+1)B_{n-1}-1]}$, and similarly for R.

Claim 9.1. Let $(j,s) \in [0, Q_{n-1}-1] \times [Q_{n-1}, U_{n-1}-2Q_{n-1}]$. Then the rectangles $P^{(sM_{n-1})}_{(jM_{n-1})}$ and $R^{(sM_{n-1})}_{(jM_{n-1})}$ are non-blank macro-cells of the simulation S_{n-1}.

Proof of claim. First, we have $T = \zeta^{[0,U_{n-1}-1]}_{[0,Q_{n-1}-1]} \in C^u_{n-1}$, since P is mapped to T by the state function $\pi^s_{n-1} \circ \pi_{n-1}$. Lemma 7.2 implies that the cell ζ^s_j is consistent with the macro-cell T. But then the square pattern $T' = \xi^{[sM_{n-1},(s+1)M_{n-1}-1]}_{[jM_{n-1},(j+1)M_{n-1}-1]}$ is a macro-cell of the sparse simulation S^s_{n-1}, since it is mapped to ζ^s_j by the state function π^s_{n-1}. This implies that $\xi^{sM_{n-1}}_{jM_{n-1}}$ is the base of the macro-cell T'. We have $\xi^{sM_{n-1}}_{jM_{n-1}} = \pi_{n-1}(P^{(sM_{n-1})}_{(jM_{n-1})})$ by definition of ξ, implying that $P^{(sM_{n-1})}_{(jM_{n-1})}$ is a non-blank macro-cell of S_{n-1}. The claim for $R^{(sM_{n-1})}_{(jM_{n-1})}$ is proved analogously, after translating R to the origin. □

Consider now the trajectory ζ of $\hat{\Psi}_{n-1}$. From Lemma 7.2 we in particular deduce that the vertical column $\zeta^{[Q_{n-1}+1,U_{n-1}-2Q_{n-1}]}_{\frac{3}{4}Q_{n-1}}$ consists of 0-demanding cells. Since the sparse simulation π^s_{n-1} is connecting by Lemma 6.3, this implies that in the trajectory ξ, there is a chain $(\xi^s_{j_s})^{U'_{n-1}-2Q'_{n-1}}_{s=Q'_{n-1}}$ of non-blank cells such that $j_{mM_{n-1}} = \frac{3}{4}Q'_{n-1}$ for all $m \in [Q_{n-1}, U_{n-1} - 2Q_{n-1}]$, and $j_{s-1} = j_s + \delta$ for some $\delta \in \{-1, 0, 1\}$ for each s. This chain corresponds, via the state function π_{n-1}, to a chain $(P^{(s)}_{(j_s)})^{U'_{n-1}-2Q'_{n-1}}_{s=Q'_{n-1}}$ of non-blank macro-cells of S_{n-1} in the trajectory η. Similarly, there is a chain of non-blank macro-cells $(R^{(s)}_{(k_s)})^{U'_{n-1}-2Q'_{n-1}}_{s=Q'_{n-1}}$ such that $k_{mM_{n-1}} = 0$ for all $m \in [2Q_{n-1}, U_{n-1} - 2Q_{n-1}]$, but $k_{Q'_{n-1}} = Q'_{n-1} - 1$. These chains are represented by dotted lines on the left hand side of Figure 6, and the macro-cells $P^{(s)}_{(s_j)}$ and $R^{(s)}_{(k_s)}$ are represented by the gray rectangles on the right hand side.

Claim 9.2. Let $s \in [Q'_{n-1}, U'_{n-1} - 2Q'_{n-1}]$. If $n \geq 2$, then the rectangles $P^{(s-1)}_{(j_s)}$ and $R^{(s-1)}_{(k_s)}$ under the macro-cells of the aforementioned chains are themselves macro-cells of the simulation S_{n-1}.

These lower rectangles are represented by the white rectangles on the right hand side of Figure 6.

Proof of claim. First, the rectangle $T'' = \theta^{[sU_{n-2},(s+1)U_{n-2}-1]}_{[j_sQ_{n-2},(j_s+1)Q_{n-2}-1]}$ is a macro-cell of the universal simulation S^u_{n-2} in a non-blank state, since it is mapped to the non-blank cell $\xi^s_{j_s}$ by the state function π^u_{n-2}. By the first item of Lemma 7.2, the rectangle $\theta^{[(s-1)U_{n-2},U_{n-2}-1]}_{[j_sQ_{n-2},(j_s+1)Q_{n-2}-1]}$ under T'' is also a macro-cell of S^u_{n-2}. But

these cells are the $\pi^s_{n-2} \circ \pi_{n-2}$-images of the corresponding rectangles $P^{(s-1)}_{(j_s)}$, which are then macro-cells of S_{n-1} (not necessarily in a non-blank state). The case of the $R^{(s-1)}_{(k_s)}$ is again similar. □

Since the rectangle $R^{(3Q'_{n-1})}_{(k_{3Q'_{n-1}})}$ lies to the left of every $P^{(s)}_{(j_s)}$ but $R^{(2Q'_{n-1})}_{(k_{2Q'_{n-1}})}$ does not, there are some $s, r \in [Q'_{n-1}, U' - 2Q'_{n-1}]$ and $\delta, \epsilon \in \{0, 1\}$ such that the rectangles $P^{(s-\delta)}_{(j_s)}$ and $R^{(r-\epsilon)}_{(k_r)}$ overlap by more than $\frac{1}{3}B_{n-1}$ cells horizontally and $\frac{1}{3}W_{n-1}$ cells vertically. If $n \geq 2$, both rectangles are macro-cells of the simulation S_{n-1}, and the induction hypothesis implies that $P^{(s-\delta)}_{(j_s)} = R^{(r-\epsilon)}_{(k_r)}$. Thus i is divisible by B_{n-1} and t by W_{n-1}, and we denote $i' = i/B_{n-1}$ and $t' = t/W_{n-1}$. If $n = 1$, we have $B_{n-1} = W_{n-1} = 1$, so the divisibility property holds trivially.

Now, the rectangle $R' = \xi^{[t',t'+U'_{n-1}]}_{[i',i'+Q'_{n-1}]}$ is mapped to R by the state function π_{n-1}, and thus is a macro-cell of the simulation S'_{n-1} of Φ_n by Φ_{n-1}. Similarly, we have $P' = \xi^{[0,0+U'_{n-1}]}_{[0,0+Q'_{n-1}]} \in C'_{n-1}$. The two chains $(\xi^s_{j_s})^{U'_{n-1}-2Q'_{n-1}}_{s=Q'_{n-1}}$ and $(\xi^{t'+s}_{i'+k_s})^{U'_{n-1}-2Q'_{n-1}}_{s=Q'_{n-1}}$ cross each other in the trajectory ξ, so that the topmost cells $\xi^{U'_{n-1}-2Q'_{n-1}}_0$ and $\xi^{t'+U'_{n-1}-2Q'_{n-1}}_{i'}$ on the left borders of P' and R' are weakly connected. Since the two cells are bases of macro-cells of the sparse simulation π^s_{n-1}, Lemma 6.6 implies that i' and t' are both divisible by M_{n-1}, and we denote $i'' = i'/M_{n-1}$ and $t'' = t'/M_{n-1}$. Then the rectangles $R'' = \zeta^{[t'',t''+U_{n-1}-1]}_{[i'',i''+Q_{n-1}-1]}$ and $P'' = \zeta^{[0,U_{n-1}-1]}_{[0,Q_{n-1}-1]}$ are macro-cells of the universal simulation S^u_{n-1}. Finally, we have $i'' \leq \frac{2}{3}Q_{n-1}$ and $t'' \leq \frac{2}{3}U_{n-1}$ by assumption, so the claim follows from Corollary 7.1. □

Finally, we claim that a non-blank macro-cell of any level cannot be very far from a non-blank macro-cell of a higher level. The result follows from the rigidity and connectivity of our simulations (Lemma 9.1), together with Lemma 6.4 and Lemma 7.5, which proved the analogous results for the sparse and universal simulations.

Lemma 9.3. *Let $n \in \mathbb{N}$, and let $\eta \in \tilde{\mathcal{T}}_{\Phi_0}$. Suppose that $P = \eta^{[t,t+W_n-1]}_{[i,i+B_n-1]}$ is a non-blank macro-cell of S_n for some $(i, t) \in \mathbb{Z}^2$. Then there exist coordinates $j \in [i - 3B_{n+1}, i + 2B_{n+1}]$ and $s \in [t - 2W_{n+1}, t - W_{n+1} + 1]$ such that $R = \eta^{[s,s+W_{n+1}-1]}_{[j,j+B_{n+1}-1]}$ is a non-blank macro-cell of S_{n+1} and the base of P is directly connected to that of R. Furthermore, B_n divides $|i - j|$ and W_n divides $|t - s|$.*

By the above and a simple induction we obtain the following lemma, which states that non-blank cells are found only in the vicinity of correct macro-cells.

Corollary 9.1. *Let $n \in \mathbb{N}$, and let $\eta \in \tilde{\mathcal{T}}_{\Phi_0}$. Suppose that $\eta^i_t \neq B_{\Gamma_0}$ for some $(i, t) \in \mathbb{Z}^2$. Then there exist coordinates $j \in [i-4B_n, i+3B_n]$ and $s \in [t-3W_n, t]$ such that $P = \eta^{[s,s+W_n-1]}_{[j,j+B_n-1]}$ is a non-blank macro-cell of S_n.*

9.2. Unique Ergodicity

We are now ready to prove our main result, the unique ergodicity of Φ_0, in the case that the effective set N is infinite.

Theorem 9.1. *The cellular automaton $\Phi_0 = G_s^N(0, \Psi_0)$ is not nilpotent. It is uniquely ergodic if and only if N is infinite.*

Proof. First, we need to show that Φ_0 is not nilpotent. Let $n \in \mathbb{N}$, and let $\eta \in \mathcal{T}^+_{\Psi_n}$ be a one-directional trajectory of Ψ_n such that the central cell η_0^0 is not blank. By the definition of the simulation S_n, there exists $\xi \in \mathcal{T}^+_{\Psi_0}$ such that $\pi_n(\xi) = \eta$. This implies that $\xi_{[0,B_n-1]}^{[0,W_n-1]} \in C_n$, and by the definition of C_n, the base cell $\xi_0^{W_n-2B_n}$ is not blank. Since $W_n - 2B_n$ grows arbitrarily large with n, the CA Ψ_0 is not nilpotent by definition.

Second, if there exists $m \in \mathbb{N}$ such that $n \notin N$ for all $n \geq m$, then there is a trajectory $\eta \in \mathcal{T}^+_{\Phi_m}$ such that $\eta_i^t \neq B_{\Gamma_n}$ for all $(i,t) \in \mathbb{Z} \times \mathbb{N}$. Then for the trajectory $\xi \in \mathcal{T}^+_{\Phi_0}$ with $\pi_n(\xi) = \eta$ we clearly have

$$\limsup_{s \longrightarrow \infty} \frac{1}{s} \left| \left\{ t \in [0, s-1] \;\middle|\; \xi_0^t \neq B_{\Gamma_0} \right\} \right| \geq W_m^{-1} > 0.$$

Corollary 3.1 now shows that Φ_0 is not uniquely ergodic.

Finally, suppose that N is infinite, and let $\eta \in \widetilde{\mathcal{T}}_{\Phi_0}$ be a two-directional trajectory. For $n \in \mathbb{N}$, we say that two macro-cells $\eta_{[i,i+B_n-1]}^{[t,t+W_n-1]}$ and $\eta_{[j,j+B_n-1]}^{[s,s+W_n-1]}$ of S_n are *spatio-temporally consistent* if $i \equiv j \bmod B_{n-1}$ and $t \equiv s \bmod W_{n-1}$. We first use the pigeonhole principle to prove the following auxiliary result.

Claim 9.3. There exists a number $q \in \mathbb{N}$ with the following property: Let $n \geq 1$, and let $K \subset [-4B_{n-1}, 3B_{n-1}] \times [-W_n, 2W_n - 1]$ be such that for all $(i,t) \in K$, the rectangle $P_{(i)}^{(t)} = \eta_{[i,i+B_{n-1}-1]}^{[t,t+W_{n-1}-1]}$ is a non-blank macro-cell of the simulation S_{n-1}, and for all distinct $(i,t), (j,s) \in K$, $P_{(i)}^{(t)}$ and $P_{(j)}^{(s)}$ are spatio-temporally inconsistent. Then we have $|K| \leq q$.

Proof of claim. First, Lemma 9.3 implies that for any rectangle $P_{(i)}^{(t)}$ for $(i,t) \in [-4B_{n-1}, 3B_{n-1} - 1] \times [-W_n, 2W_n - 1]$ which is a non-blank macro-cell of S_{n-1}, there exist $j \in [-4(Q'_n + 1)B_{n-1}, (3Q'_n + 2)B_{n-1}]$ and $s \in [-3W_n, W_n]$ such that $R_{(j)}^{(s)} = \eta_{[j,j+B_n-1]}^{[s,s+W_n-1]}$ is a non-blank macro-cell of S_n, and $i \equiv j \bmod B_{n-1}$ and $t \equiv s \bmod W_{n-1}$ hold. We associate one such pair (j,s) to (i,t), and denote $(j,s) = F(i,t)$. Note that we in particular have $j \in [-5B_n, 4B_n]$ in the above. Lemma 9.2, together with a pigeonhole argument, then implies that there exists $q \in \mathbb{N}$ independent of n such that at most q of the rectangles $R_{(j)}^{(s)}$ for $(j,s) \in [-5B_n, 4B_n] \times [-3W_n, W_n]$ are macro-cells of S_{n+1}. Thus, the function $F : K \to [-5B_n, 4B_n] \times [-3W_n, W_n]$ has an image of size at most q. Finally, if $F(i,t) = F(i',t')$ for some $(i,t), (i',t') \in K$, then the bases macro-cells $P_{(i)}^{(t)}$ and $P_{(i')}^{(t')}$ are spatio-temporally consistent, so that $(i,t) = (i',t')$ by assumption. Thus F is injective, which implies $|K| \geq q$. □

Claim 9.4. Let $n \in \mathbb{N}$ be such that $n+1 \in N$, let $\ell \in \mathbb{N}$, and denote

$$L = \{t \in [\ell W_{n+1}, (\ell+1)W_{n+1} - 1] \mid \eta_0^t \neq B_{\Gamma_0}\}.$$

Then we have $|L| \leq \frac{48qW_{n+1}}{n}$.

Proof of claim. Denote $A = [-4B_n, 3B_n] \times [\ell W_{n+1} - 3W_n, (\ell+1)W_{n+1} - 1]$, and let $(j, s) \in A$ be such that $P = \eta_{[j,j+B_n-1]}^{[s,s+W_n-1]}$ is a non-blank macro-cell of S_n. Without loss of generality, assume that j is divisible by B_n and s by W_n, and let $\xi = \pi_n(\eta)$. By Lemma 9.1 we have $\xi \in \widetilde{\mathcal{T}}_{\Phi_n}$. Then, denoting $j' = jB_n^{-1}$ and $s' = sW_n^{-1}$, we have $\xi_{j'}^{s'} \neq B_{\Gamma_n}$. By Lemma 6.5, the set L' of those coordinates $(k', r') \in [-3, 2] \times [\ell U'_{n+1} - 3, (\ell+1)U'_{n+1} - 1]$ for which $\xi_{k'}^{r'}$ is connected to $\xi_{j'}^{s'}$ has size at most $\frac{15}{n}(U'_{n+1} + 3)$. Now, if $\eta_{[r,r+B_n-1]}^{[k,k+W_n-1]}$ is a non-blank macro-cell of S_n which is spatio-temporally consistent with P, then $k' = kB_n^{-1}$ and $r' = rW_n^{-1}$ are both integers, and hence $(k', r') \in L'$. Thus the number of such coordinates (k, r) is also at most $\frac{15}{n}(U'_{n+1} + 3)$.

Denote by H the set of those $(j, s) \in A$ for which $\eta_{[j,j+B_n-1]}^{[s,s+W_n-1]}$ is a non-blank macro-cell of S_n. By Claim 9.3, there are at most q classes of spatio-temporally consistent macro-cells in A, so the cardinality of H is at most $\frac{15q}{n}(U'_{n+1} + 3)$.

Let now $t \in L$. By Corollary 9.1, there exists $(j, s) \in [-4B_n, 3B_n] \times [t - 3W_n, t]$ such that $\eta_{[j,j+B_n-1]}^{[s,s+W_n-1]}$ is a non-blank macro-cell of S_n, that is, $(j, s) \in H$. Hence $L \subset \{t \in [s, s + 3W_n] \mid (j, s) \in H\}$, implying that

$$|L| \leq 3W_n|H| \leq \frac{45qW_n}{n}(U'_{n+1} + 3) \leq \frac{48qW_{n+1}}{n},$$

since $W_n \leq W_{n+1}$. □

Since the effective set N contains arbitrarily large numbers, the above claim directly implies that

$$\limsup_{s\to\infty} \frac{1}{n} \left|\left\{t \in [0, s-1] \;\middle|\; \eta_0^t \neq B_{\Gamma_0}\right\}\right| = 0,$$

and Corollary 3.1 then shows that Φ_0 is uniquely ergodic. □

10. Further Results

In this section, we explore some extensions and variants of Theorem 9.1 that can be proved using an amplifier construction. We also state some undecidability results regarding them.

10.1. Asymptotic Nilpotency in Density

The product topology, induced by the Cantor metric, is usually seen as the natural environment for cellular automata. However, since it places a heavy emphasis on the central coordinates of a configuration, more 'global' topologies have been defined, one of the most well-known being the Besicovitch topology.

Definition 10.1. The *Besicovitch pseudometric* $d_B : \Sigma^{\mathbb{Z}} \times \Sigma^{\mathbb{Z}} \to \mathbb{R}$ is defined by

$$d_B(x, y) = \limsup_{n \longrightarrow \infty} \frac{1}{2n+1} |\{i \in [-n, n] \mid x_i \neq y_i\}|$$

for $x, y \in \Sigma^{\mathbb{Z}}$. The *Besicovitch class* $[x]$ of a configuration $x \in \Sigma^{\mathbb{Z}}$ is the set $\{y \in \Sigma^{\mathbb{Z}} \mid d_B(x, y) = 0\}$.

The function d_B is a pseudometric on $\Sigma^{\mathbb{Z}}$, and thus induces a metric on the set $\{[x] \mid x \in \Sigma^{\mathbb{Z}}\}$ of Besicovitch classes of configurations. It is known that this metric space it complete, but not compact [1]. Instead of comparing the central coordinates, it measures the asymptotic density of the set of differing cells. The Besicovitch pseudometric was first studied in the context of cellular automata in [3].

The following result appeared in [19], and we repeat the proof here for completeness.

Proposition 10.1. *Let $\Psi : \Sigma^{\mathbb{Z}} \to \Sigma^{\mathbb{Z}}$ be a cellular automaton such that for all $x \in \Sigma^{\mathbb{Z}}$ there exists $n \in \mathbb{N}$ with $\Psi^n(x) \in [{}^{\infty}B_{\Sigma}^{\infty}]$. Then Ψ is nilpotent.*

Proof. Let $x \in \Sigma^{\mathbb{Z}}$ be a generic point for the uniform Bernoulli distribution on Σ (see [5]). This means that

$$\lim_{n \to \infty} \frac{1}{2n+1} \left|\left\{i \in [-n, n] \;\middle|\; x_{[i,i+|w|-1]} = w\right\}\right| = |\Sigma|^{-|w|} \tag{5}$$

holds for all $w \in \Sigma^*$. By assumption, $\Psi^n(x) \in [{}^{\infty}B_{\Sigma}^{\infty}]$ holds for some $n \in \mathbb{N}$. Let $w \in \Sigma^{2n+1}$ be arbitrary. If we had $\Psi^n(w) \neq B_{\Sigma}$, then $d_B(\Psi^n(x), {}^{\infty}B_{\Sigma}^{\infty}) \geq |\Sigma|^{-2n-1}$ by (5), a contradiction. Thus we have $\Psi^n(w) = B_{\Sigma}$, and since $w \in \Sigma^{2n+1}$ was arbitrary, we have $\Psi^n(\Sigma^{\mathbb{Z}}) = \{{}^{\infty}B_{\Sigma}^{\infty}\}$. □

In particular, the above holds if $\Psi^n(x)$ actually equals ${}^{\infty}B_{\Sigma}^{\infty}$. It is also well known (see for instance [4]) that if for all $\epsilon > 0$ there exists $n \in \mathbb{N}$ such that $d_C(\Psi^n(x), {}^{\infty}B_{\Sigma}^{\infty}) < \epsilon$ for all $x \in \Sigma^{\mathbb{Z}}$, then $\Omega_{\Psi} = \{{}^{\infty}B_{\Sigma}^{\infty}\}$, and thus Ψ is nilpotent by Proposition 3.1. Next, we show that the Besicovitch analogue of this result does not hold, and in fact the automaton Φ_0 from the sparse amplifier is a counterexample. First, we give a name for the property.

Definition 10.2. Let $\Psi : \Sigma^{\mathbb{Z}} \to \Sigma^{\mathbb{Z}}$ be a cellular automaton. Suppose that for all $\epsilon > 0$ there exists $n \in \mathbb{N}$ such that $d_B(\Psi^n(x), {}^{\infty}B_{\Sigma}^{\infty}) < \epsilon$ holds for all $x \in \Sigma^{\mathbb{Z}}$. Then we say Ψ is *asymptotically nilpotent in density* (AND for short).

In other words, a cellular automaton is asymptotically nilpotent in density if the asymptotic density of non-blank cells in a configuration converges to 0 under the action of the CA, and the speed of the convergence is uniform. The condition of uniform convergence can be dropped, as shown by the following characterization.

Lemma 10.1. *Let $\Psi : \Sigma^{\mathbb{Z}} \to \Sigma^{\mathbb{Z}}$ be a cellular automaton. Then Ψ is asymptotically nilpotent in density if and only if $\Omega_\Psi \subset [^\infty B_\Sigma^\infty]$, if and only if*

$$\lim_{\ell \longrightarrow \infty} \max_{w \in \mathcal{B}_\ell(\Omega_\Psi)} \frac{1}{\ell} |\{i \in [0, \ell - 1] \mid w_i \neq B_\Sigma\}| = 0. \tag{6}$$

Proof. First, if Ψ is AND, it is clear that $\Omega_\Psi \subset [^\infty B_\Sigma^\infty]$ holds. Suppose then that (6) holds, and let $\epsilon > 0$. Then there exists a length $\ell \in \mathbb{N}$ such that $|\{i \in [0, \ell - 1] \mid w_i \neq B_\Sigma\}| < \epsilon \cdot \ell$ holds for all $w \in \mathcal{B}_\ell(\Omega_\Psi)$. It is known (see again [4]) that there now exists $n \in \mathbb{N}$ such that $\mathcal{B}_\ell(\Psi^n(\Sigma^{\mathbb{Z}})) = \mathcal{B}_\ell(\Omega_\Psi)$. But this implies that $d_B(\Psi^n(x), {}^\infty B_\Sigma^\infty) < \epsilon$ for all $x \in \Sigma^{\mathbb{Z}}$, and since ϵ was arbitrary, Ψ is AND.

We still need to show that $\Omega_\Psi \subset [^\infty B_\Sigma^\infty]$ implies (6), and for that, consider the shift map $\sigma : \Omega_\Psi \to \Omega_\Psi$, which is a cellular automaton on Ω_Ψ. Using the notation of Section 3, we have $d_\sigma(B_\Sigma, x) = d_{\sigma^{-1}}(B_\Sigma, x) = 1$ for all $x \in \Omega_\Psi$, and Proposition 3.2 implies that the convergence of the limits is uniform in x. But this is equivalent to (6), and we are done. □

We only sketch the proof of the following result, since it is mostly analogous to that of Theorem 9.1.

Proposition 10.2. *The automaton Φ_0 is asymptotically nilpotent in density if and only if the effective set N is infinite.*

Proof sketch. If N is finite, then Φ_0 is not AND, for the same reason as in Theorem 9.1.

Suppose now that N is infinite. By Lemma 10.1, it suffices to prove that $\Omega_{\Phi_0} \subset [^\infty B_{\Gamma_0}^\infty]$, so let $x \in \Omega_{\Phi_0}$ be arbitrary. Then there exists a two-directional trajectory $\eta \in \mathcal{T}_{\Phi_0}$ with $\eta^0 = x$. It is easy to see that the variant of Lemma 6.5, where η_0^s is replaced by η_s^0, holds. We can also prove the variant of Claim 9.3 in the proof of Theorem 9.1 where $K \subset [-B_n, B_n] \times [-3W_{n-1}, W_{n-1} - 1]$. Finally, we can prove a variant of Claim 9.4 where

$$L = \{i \in [\ell B_{n+1}, (\ell + 1)B_{n+1} - 1] \mid \eta_i^0 \neq B_{\Gamma_0}\},$$

and the bound is modified accordingly. This shows that $x = \eta^0 \in [^\infty B_{\Gamma_0}^\infty]$, so that we have $\Omega_{\Phi_0} \subset [^\infty B_{\Gamma_0}^\infty]$. □

The main result of [9] states that if a cellular automaton $\Psi : \Sigma^{\mathbb{Z}} \to \Sigma^{\mathbb{Z}}$ satisfies $\Psi^n(x) \overset{n \to \infty}{\longrightarrow} {}^\infty B_\Sigma^\infty$ for all $x \in \Sigma^{\mathbb{Z}}$ with respect to the Cantor metric d_C, then it is nilpotent. The above proposition in particular shows that the d_B-analogue of this result does not hold. In the same article, it was asked whether the condition that the d_C-orbit closure $\overline{\{\Psi^n(x) \mid n \in \mathbb{N}\}}$ or every $x \in \Sigma^{\mathbb{Z}}$ contains the uniform configuration ${}^\infty B_\Sigma^\infty$ implies nilpotency, and Theorem 9.1 clearly disproves this. Finally, [10, Corollary 31] states that if the limit set Ω_Ψ contains only B_Σ-finite configurations, then Ψ is nilpotent. Lemma 10.1 and Proposition 10.2 show that this result is strict, in the sense that Ω_Ψ containing

only configurations where the asymptotic density of blank symbols is 1 is not enough to guarantee nilpotency.

Next, we show that Proposition 10.2 is 'only' an artifact of our construction, since the notions of unique ergodicity and AND are independent. For this, we make two modifications to the sparsification transformation G_s. More explicitly, we define two new transformations G_1 and G_2 that add some new fields and rules to their input program, and replace G_s with either $G_1 \circ G_s$ or $G_2 \circ G_s$. When defining G_1 and G_2, we thus assume that the input program is already sparsified by G_s, and thus contains a numeric parameter `N` and a numeric field `Cnt`. These transformations rename the fields, parameters and procedures of the input program if necessary, except for those defined by G_s, which they use during their own execution.

First, the application of G_1 to $n \in \mathbb{N}$ and a sparsified program p is defined in Algorithm 8. The transformation G_1 adds one new counter, `Cnt2`, ranging from 0 to $2n$, that is initialized to 1 at each base cell (a right-moving cell with $\texttt{Cnt} = 0$). The counter field is then incremented by one for $2n$ steps, after which it remains at 0, until the cell becomes a base cell again. The new counter operates on a different 'layer' as the sparsified automaton, and does not affect its dynamics. Geometrically, G_1 adds to all trajectories of G_s vertical lines that connect each base cell to the one above it, as shown in Figure 8.

Algorithm 8 The transformed program $G_1(n, p)$.

```
num field Cnt2 ≤ 2N
p                                          ▷ Definitions and body of p
if Kind = ⇒ ∧ Cnt = 0 then
    Cnt2 ← 1                   ▷ The base of a macro-cell stays non-blank...
else if Cnt2 > 0 then
    Cnt2 ← Cnt2 + 1 mod 2N                 ▷ ... for exactly 2N steps
else
    Cnt2 ← 0
```

In the second transformation, G_2, we wish to introduce horizontal lines instead of vertical ones. This is a little more complex, and to define G_2, we need the following auxiliary concepts.

Definition 10.3. We inductively define a sequence $(L(n))_{n\geq 1}$ of finite sets $L(n) \subset \mathbb{Z}^2$ as follows. First, let $L(1) = \{(0,0)\}$ and $L(2) = \{(0,0), (-1,1), (0,1), (1,1)\}$. For $n \geq 3$, we denote $n^* = \lceil \frac{n+1}{2} \rceil$ and $n_* = \lfloor \frac{n+1}{2} \rfloor$, and define

$$L(n) = \{(k, \pm k) \mid 0 \leq k \leq n^*\} \cup L(n^*) + (n_*, -n_*) \cup L(n_*) + (n^*, n^*).$$

For each $n \geq 1$, define also a configuration $\eta_n \in (L(n) \cup \{\#\})^{\mathbb{Z}^2}$ by

$$(\eta_n)_i^t = \begin{cases} (i,t), & \text{if } (i,t) \in L(n), \\ \#, & \text{otherwise,} \end{cases}$$

and a function $f_n : (L(n) \cup \{\#\})^3 \to L(n) \cup \{\#\}$ by

$$f_n(\vec{u}, \vec{v}, \vec{w}) = \begin{cases} (i,t) \in L(n) \setminus \{\vec{0}\}, & \text{if } (\vec{u}, \vec{v}, \vec{w}) = g_n(\eta_n)^{t-1}_{[i-1,i+1]}, \\ \#, & \text{otherwise.} \end{cases}$$

The sets $L(n)$ and the functions f_n satisfy the following properties, which are not hard to prove:

1. $[-n+1, n-1] \times \{n-1\} \subset L(n) \subset [-n+1, n-1] \times [0, n-1]$,
2. the cardinality of the intersection of $L(n)$ with any vertical column is logarithmic in n,
3. the sets $L(n)$ and functions f_n are computable in polynomial time uniformly in n, and
4. the restriction $g_n(\eta_n)^{[0,\infty)}$ to the upper half-plane is a one-directional trajectory of the cellular automaton with local function f_n.

See Figure 7 for a visualization of $L(n)$ for different values of n.

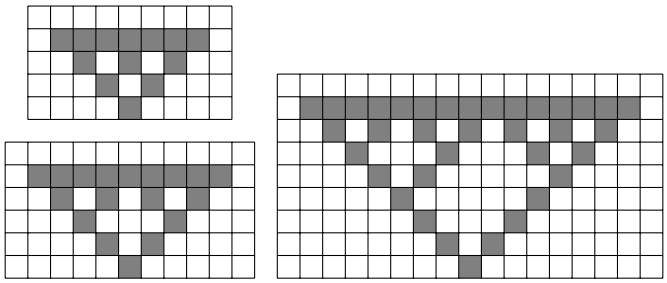

Figure 7: The sets $L(4)$ (top left), $L(5)$ (bottom left) and $L(7)$ (right). Note how the two smaller sets are used to construct $L(7)$.

The application of the second transformation G_2 to $n \in \mathbb{N}$ and p is defined in Algorithm 9. Geometrically, the idea is to draw on every macro-cell of the level-n sparse simulation a copy of the pattern $L(n)$, rooted at the base cell. This is achieved by the new field `Elem` with values in $L(\mathtt{N}) \cup \{\#\}$, of which # is presented by zeros, and the functions f_n. The field `Elem` of a base cell gets the value $(0,0) \in L(\mathtt{N})$, and on subsequent steps, the new value of `Elem` is computed using $f_{\mathtt{N}}$. See Figure 8 for a visualization.

Algorithm 9 The transformed program $G_2(n, p)$.

```
1: enum field Elem ∈ L(N) ∪ {#}
2: p                                          ▷ Definitions and body of p
3: if Kind = ⇒ ∧ Cnt = 0 then
4:     Elem ← (0, 0)                          ▷ From the base of a macro-cell...
5: Elem ← f_N(Elem⁻, Elem, Elem⁺)             ▷ ...construct the set L(N)
```

Now, let $N \subset \mathbb{N}$ be decidable in polynomial time, and consider the transformations

$$G_i^N(n,p) = \begin{cases} G_s(n,p), & \text{if } n \in N, \\ (G_i \circ G_s)(n,p), & \text{if } n \notin N \end{cases}$$

for $i \in \{1,2\}$. Let the cellular automata $\Theta_i : \Delta_i^{\mathbb{Z}} \to \Delta_i^{\mathbb{Z}}$ be the bottom automata of the amplifier sequences obtained by applying Theorem 8.1 to G_i^N.

Proposition 10.3. *The cellular automaton Θ_1 (Θ_2) is AND (uniquely ergodic), and it is uniquely ergodic (AND, respectively) if and only if the effective set N is infinite. In particular, the notions of unique ergodicity and AND are independent.*

Proof sketch. Consider first the automaton Θ_1. It is easy to see that the proof of Proposition 10.2 works with Φ_0 replaced by Θ_1, since all the results of Section 6.2 also hold for G_1, with Lemma 6.5 replaced by the horizontal version mentioned in the proof of Proposition 10.2, with a slightly weaker bound. If the effective set N is infinite, the CA is uniquely ergodic, since the proof of Theorem 9.1 holds almost as such. However, if $N \subset [0, n-1]$ for some $n \in \mathbb{N}$, the n'th automaton in the amplifier given by Theorem 8.1 for G_1^N has a trajectory whose central column contains no blank cells, and then Θ_1 is not uniquely ergodic.

The case of Θ_2 is similar: All results of Section 6.2 hold for G_2, with a weaker bound in Lemma 6.5 caused by the new cells with `Elem`-value different from #, and thus Θ_2 is uniquely ergodic by the same proof as Φ_0. If N is infinite, it is also AND for the same reason as Φ_0, and if N is finite, then some automaton in the amplifier of G_2^N has a trajectory whose central horizontal line contains no blank cells, implying that Θ_2 is not AND. □

10.2. Asymptotic Sparsity in Other Directions

As unique ergodicity corresponds to every column of every trajectory being asymptotically sparse, the AND property is its analogue for the horizontal direction. As a generalization of this idea, we sketch the proof of the analogous result for every rational direction. Recall that $\sigma : \Gamma_0^{\mathbb{Z}} \to \Gamma_0^{\mathbb{Z}}$ is the shift map, and note that the CA $\sigma^i \circ \Phi_0^k$ has a larger radius than 1, if $i \neq 0$ or $k > 1$.

Proposition 10.4. *Let $i \in \mathbb{Z}$ and $k > 0$. Then the cellular automaton $\sigma^i \circ \Phi_0^k$ is uniquely ergodic if and only if N is infinite.*

Proof sketch. The case of finite N is shown as in Theorem 9.1, so suppose that N is infinite. First, for all $a \in \mathbb{Q}$ such that $a \notin \{-1, 1\}$, we can prove the generalization of Lemma 6.5 where η_0^s is replaced by $\eta_s^{\lfloor as \rfloor}$, with the bound depending on a, but approaching the original as a approaches 0. Then, we again adapt the proof of Theorem 9.1. This time, we need to show that the asymptotic density of non-blank cells on the line $I = \{\eta_{mi}^{mk} \mid m \in \mathbb{N}\}$ is 0. For this, we prove the analogue of the first claim, where $K \subset \bigcup_{s \in [0, W_n - 1]} [si - 4B_{n-1}, si + 3B_{n-1}] \times [sk - 3W_{n-1}, sk]$, with the bound $q \in \mathbb{N}$ depending on i and k.

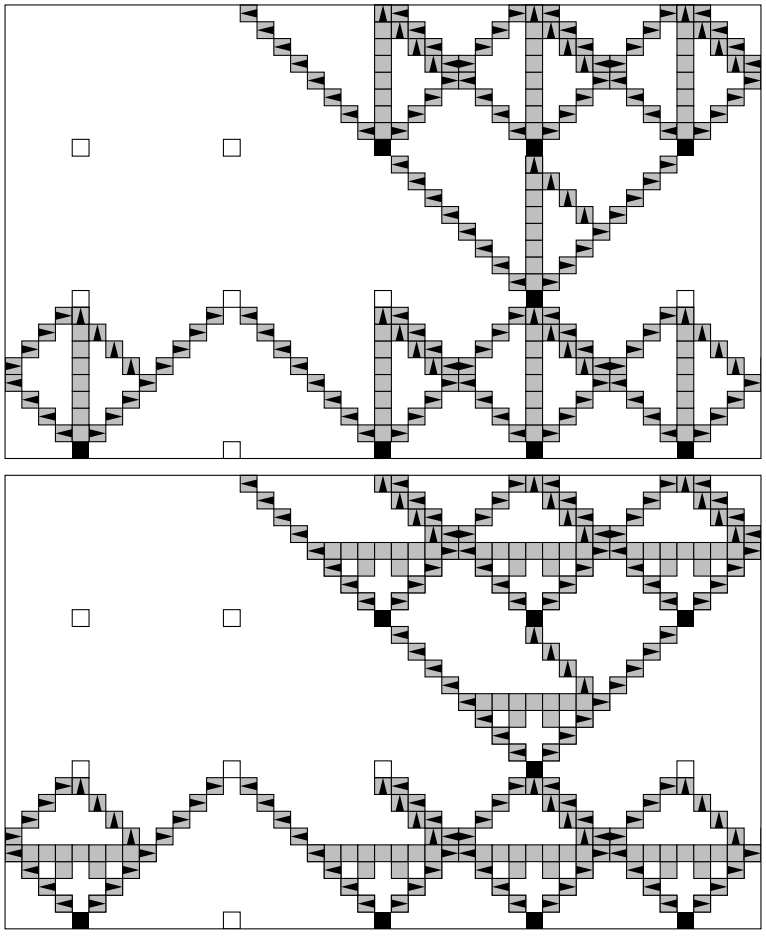

Figure 8: The two modifications of G_s, applied to the three-neighbor XOR automaton with $n = 4$, with G_1 on top and G_2 below. The unmarked cells are not live, but have nonzero `Cnt2`-values and `Elem`-values, respectively. Compare with Figure 2.

Consider then the variant of the second claim where we have redefined $L = \{t \in [\ell W_{n+1}, (\ell+1)W_{n+1} - 1] \mid \eta_{ti}^{tk} \neq B_{\Gamma_0}\}$. In its proof, note that the set of coordinates $(s, j) \in \mathbb{Z}^2$ such that $I \cap [j, j + B_n - 1] \times [s, s + W_n - 1] \neq \emptyset$ approximates the discrete line $J = \{(r, \lfloor \frac{kW_n}{iB_n} r \rfloor) \mid r \in \mathbb{N}\}$. Then the generalization of Lemma 6.5 we proved earlier, together with the fact that $\frac{kW_n}{iB_n}$ converges to 0 as n grows, allows us to prove an upper bound for $|L|$, which approaches $\frac{EqW_{n+1}}{n}$ for some constant $E \in \mathbb{N}$ (depending on i and k) as n grows. The proof is finished as in Theorem 9.1. □

10.3. Computability

In this section, we turn to decision problems concerning unique ergodicity and the AND property. We start by proving their undecidability, which is the reason for defining the effective set N. Note that a Π_2^0-complete problem is in particular undecidable.

Proposition 10.5. *It is Π_2^0-complete whether a given cellular automaton, or a given AND cellular automaton, is uniquely ergodic. Similarly, it is Π_2^0-complete whether a given cellular automaton, or a given uniquely ergodic cellular automaton, is AND.*

Proof. First, let M be a Turing machine, and let q be a state of M. Let $N \subset \mathbb{N}$ be the set of numbers n such that M, when run on empty input, enters the state q on its the n'th step. Then N is decidable in polynomial time.

Consider first the problem of determining whether a given CA is uniquely ergodic. Apply Theorem 8.1 to the partial sparsification G_s^N, and consider the bottom automaton Φ_0. By Theorem 9.1, Φ_0 is uniquely ergodic if and only if N is infinite. Since the infinity of N, when M is given, is well known to be Π_2^0-hard, so is the unique ergodicity of a given CA. The Π_2^0-hardness of the other properties follows similarly, using the transformations G_1^N and G_2^N, together with Propositions 10.2 and 10.3.

For the other direction, Proposition 3.2 implies that the unique ergodicity of a cellular automaton $\Psi : \Sigma^{\mathbb{Z}} \to \Sigma^{\mathbb{Z}}$ is equivalent to the formula

$$\forall k \in \mathbb{N} : \exists \ell \in \mathbb{N} : \forall w \in \Sigma^{2\ell+1} : |\{\Psi^n(w)_{\ell-n-1} \neq B_\Sigma \mid n \in [0, \ell-1]\}| \leq \frac{\ell}{k},$$

while asymptotic nilpotency in density is equivalent to the formula

$$\forall k \in \mathbb{N} : \exists n, \ell \in \mathbb{N} : \forall w \in \Psi^n(\Sigma^{2(n+\ell)+1}) : |\{w_i \neq B_\Sigma \mid i \in [0, \ell-1]\}| \leq \frac{2\ell+1}{k}$$

by the proof of Lemma 10.1. Since these are both Π_2^0 formulae, the properties are Π_2^0-complete. □

It is a well known theorem, first proved in [13], that the nilpotency of a given cellular automaton is undecidable. We strengthen this result by restricting to the class of uniquely ergodic and AND cellular automata.

Proposition 10.6. *It is undecidable whether a given uniquely ergodic and AND cellular automaton is nilpotent.*

Proof. Let M be a Turing machine, and let p_0 be a program for the trivial cellular automaton on $\{0,1\}^{\mathbb{Z}}$ that sends everything to ${}^{\infty}0^{\infty}$. Define the polynomial CA transformation G_s^M by

$$G_s^M(n,p) = \begin{cases} p_0, & \text{if } M \text{ halts after } n \text{ steps on empty input,} \\ G_s(n,p), & \text{otherwise.} \end{cases}$$

Let $(\Theta_n)_{n\in\mathbb{N}}$ be the amplifier sequence of automata given by Theorem 8.1 for G_s^M. Now, if M never halts, then $G_s^M = G_s$, and Theorem 9.1 and Proposition 10.2 imply that Θ_0 is uniquely ergodic and AND, but not nilpotent. However, if M halts on the n'th step of its computation, then Θ_n is the trivial automaton, and Corollary 9.1 applied to n (which is valid up to level n) implies that Θ_0 is nilpotent. Thus we have reduced the halting problem to our restricted nilpotency problem. □

11. Conclusions and Future Directions

In this article, we have presented a uniquely ergodic cellular automaton which is not nilpotent. The construction uses a technique we call amplification, which combines self-simulation with arbitrary modification of CA rules. We have also presented some further results that are given by from minor variants of the construction, as well as related undecidability results.

Our combinatorial characterization of unique ergodicity, Corollary 3.1, only applies to cellular automata with a quiescent state. Not all cellular automata $\Psi : \Sigma^{\mathbb{Z}} \to \Sigma^{\mathbb{Z}}$ have one, but in general there are states $q_0, \dots, q_{k-1} \in \Sigma$ such that $\Psi({}^{\infty}q_i^{\infty}) = {}^{\infty}q_{i+1 \bmod k}^{\infty}$ for all $i \in [0, k-1]$, and if μ_i is the Dirac measure concentrated on ${}^{\infty}q_i^{\infty}$, then $\frac{1}{k}\sum_{i=0}^{k-1} \mu_i$ is an invariant measure of Ψ. It is likely that the characterization can be extended to this general case, and that uniquely ergodic cellular automata without a quiescent state can also be constructed. However, this seems to require some entirely new ideas, and an even more complicated automaton.

Every uniquely ergodic cellular automaton $\Phi_0 : \Gamma_0^{\mathbb{Z}} \to \Gamma_0^{\mathbb{Z}}$ with a quiescent state B_{Γ_0} also has the following property. Let $\Psi : \Sigma^{\mathbb{Z}} \to \Sigma^{\mathbb{Z}}$ be an arbitrary cellular automaton, and consider the product alphabet $\Delta = \Gamma_0 \times \Sigma$. Let $\Theta : \Delta^{\mathbb{Z}} \to \Delta^{\mathbb{Z}}$ be any CA that behaves as Φ_0 on the first component of Δ, and as Ψ on the second component of $\{B_{\Gamma_0}\} \times \Sigma$. Then we have $\mathcal{M}_\Theta = \{\mu_0\} \times \mathcal{M}_\Psi$, where μ_0 is the Dirac measure on Γ_0 concentrated on the uniform configuration ${}^{\infty}B_{\Gamma_0}^{\infty}$. In other words, Θ has essentially the same set of invariant measures as Ψ. However, if we can construct Θ so that it has some desired property, its existence proves that having that property does not essentially restrict the set of invariant measures of a CA. An obvious candidate would be (some suitable variant of) computational universality, but there are undoubtedly other properties for which the above scheme works too.

Finally, it would be interesting to modify our construction to make the resulting CA reversible. Of course, a reversible CA cannot be uniquely ergodic, since it preserves the uniform Bernoulli measure, but it might have some weaker property that would shed some light on the possible sets of invariant measures of reversible cellular automata.

Acknowledgments

I am thankful to my advisor Jarkko Kari for his advice and support, Ville Salo for many useful discussions on the topics of this article, Charalampos Zinoviadis for directing me to [7] and discussing future directions of this research, Pierre Guillon for his interest in this project, and the anonymous referees for their comments that greatly helped to improve the readability of this article.

References

[1] François Blanchard, Enrico Formenti, and Petr Kůrka. Cellular automata in the Cantor, Besicovitch, and Weyl topological spaces. *Complex Systems*, 11(2):107–123, 1997.

[2] Laurent Boyer, Victor Poupet, and Guillaume Theyssier. On the complexity of limit sets of cellular automata associated with probability measures. In *Mathematical foundations of computer science 2006*, volume 4162 of *Lecture Notes in Comput. Sci.*, pages 190–201. Springer, Berlin, 2006.

[3] Gianpiero Cattaneo, Enrico Formenti, Luciano Margara, and Jacques Mazoyer. A shift-invariant metric on $S^{\mathbb{Z}}$ inducing a non-trivial topology. In *Mathematical foundations of computer science 1997 (Bratislava)*, volume 1295 of *Lecture Notes in Comput. Sci.*, pages 179–188. Springer, Berlin, 1997.

[4] Karel Culik, II, Jan K. Pachl, and Sheng Yu. On the limit sets of cellular automata. *SIAM J. Comput.*, 18:831–842, August 1989.

[5] Manfred Denker, Christian Grillenberger, and Karl Sigmund. *Ergodic theory on compact spaces.* Lecture Notes in Mathematics, Vol. 527. Springer-Verlag, Berlin, 1976.

[6] Bruno Durand, Andrei Romashchenko, and Alexander Shen. Fixed-point tile sets and their applications. *J. Comput. System Sci.*, 78(3):731–764, 2012.

[7] Peter Gács. Reliable cellular automata with self-organization. *J. Statist. Phys.*, 103(1-2):45–267, 2001.

[8] Lawrence F. Gray. A reader's guide to P. Gács's "positive rates" paper: "Reliable cellular automata with self-organization" [J. Statist. Phys. **103** (2001), no. 1-2, 45–267; MR1828729 (2002c:82058a)]. *J. Statist. Phys.*, 103(1-2):1–44, 2001.

[9] 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.

[10] Pierre Guillon and Gaétan Richard. Asymptotic behavior of dynamical systems and cellular automata. *ArXiv e-prints*, April 2010.

[11] 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.

[12] Gustav A. Hedlund. Endomorphisms and automorphisms of the shift dynamical system. *Math. Systems Theory*, 3:320–375, 1969.

[13] Jarkko Kari. The nilpotency problem of one-dimensional cellular automata. *SIAM J. Comput.*, 21(3):571–586, 1992.

[14] Petr Kůrka and Alejandro Maass. Limit sets of cellular automata associated to probability measures. *J. Statist. Phys.*, 100(5-6):1031–1047, 2000.

[15] Shahar Mozes. Tilings, substitution systems and dynamical systems generated by them. *J. Analyse Math.*, 53:139–186, 1989.

[16] Piergiorgio Odifreddi. *Classical recursion theory*, volume 125 of *Studies in Logic and the Foundations of Mathematics*. North-Holland Publishing Co., Amsterdam, 1989. The theory of functions and sets of natural numbers, With a foreword by G. E. Sacks.

[17] Raphael M. Robinson. Undecidability and nonperiodicity for tilings of the plane. *Invent. Math.*, 12:177–209, 1971.

[18] 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*, pages 86–96. Open Publishing Association, 2012.

[19] Ilkka Törmä. Nilpotency in Cellular Automata. Master's thesis, University of Turku, Finland, 2012.

[20] Peter Walters. *An introduction to ergodic theory*, volume 79 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 1982.