For the 1998 conference on Mathematical Foundations of Computer Science (MFCS'98) four papers on Cellular Automata were accepted as regular MFCS'98 contributions. Furthermore an MFCS'98 satellite workshop on Cellular Automata was organized with ten additional talks. We are confident that the embedding of the workshop into the conference with its participants coming from a broad spectrum of fields of work will lead to interesting discussions and a fruitful exchange of ideas.

This technical report contains papers presented at the workshop. The contributions which had been accepted for MFCS'98 itself may be found in the conference proceedings, edited by L. Brim, J. Gruska and J. Zlatuška, LNCS 1450, at the following pages:

All other (invited and regular) papers of the workshop are contained in this technical report:

We would like to thank the referees of the workshop papers for their timely work, and Jozef Gruska as MFCS programme committee co-chairman and Antonin Kucera and the other organizers of MFCS hidden behind `mfcs98@fi.muni.cz` for their cooperation. They all were a great help in organizing the workshop without trouble.

Thomas Worsch and Roland Vollmar — Karlsruhe, August 1998

Cellular automata in the Cantor, Besicovitch and Weyl spaces

François Blanchard, Enrico Formenti, Petr Kůrka

July 16, 1998

Abstract

The Besicovitch and Weyl pseudometrics on the space $A^{\mathbb{Z}}$ of biinfinite sequences measure the density of differences in either the central or arbitrary segments of given sequences. The Besicovitch and Weyl spaces are obtained from $A^{\mathbb{Z}}$ by factoring through the equivalence of zero distance. We consider cellular automata as dynamical systems on the Besicovitch and Weyl spaces and compare their topological and dynamical properties with those they possess in the Cantor space.

1 Introduction

A cellular automaton consists of a biinfinite array of cells containing letters from a finite alphabet, which are updated according to a local interaction rule. Cellular automata have been of considerable interest both as models of physical and biological phenomena and in Symbolic Dynamics as homomorphisms of the shift (Hedlund [8]). They display a large spectrum of dynamical behaviours ranging from stable to chaotic dynamics and they could also support universal computation. For a survey, see Wolfram [13], Culik II, Hurd and Yu [5] or Blanchard, Maass and Kůrka [2].

When a cellular automaton is conceived as a dynamical system, the space of biinfinite sequences is equipped with the product topology, which makes it homeomorphic to the Cantor space. In this space the shift map has many chaoticity properties like sensitivity to initial conditions and topological transitivity. However, the shift may be regarded as a shift of the observation point, in which case the configuration does not change at all. To distinguish the shift from chaotic cellular automata which really change the structure of configurations, Cattaneo et al. [4] consider the shift-invariant Besicovitch pseudometric, which has been also used in the study of almost periodic functions (see e.g. Besicovitch [1]). The Besicovitch pseudometrics measures the density of differences in the central part of two given sequences. One obtains the Besicovitch space by factoring the space of biinfinite sequences by the equivalence of zero distance. A variant is

the Weyl pseudometric, which measures the density of differences in arbitrary segments of two given sequences.

Other heuristic reasons for considering the Weyl or Besicovitch space instead of the Cantor space is that the Cantor metrics gives undue importance to the coordinate 0; and also that the notion of perturbation associated to this metric cannot be interpreted physically.

Downarowicz and Iwanik [6] show that the Weyl space is pathwise connected and incomplete. The Besicovitch space is also pathwise connected but complete. Both spaces are infinite-dimensional and neither separable nor locally compact.

Cellular automata are continuous with respect to both the Besicovitch and the Weyl pseudometrics, so they define dynamical systems in the corresponding spaces. In the present paper we compare topological and dynamical properties of cellular automata in these spaces with those in the Cantor space. When passing from the Besicovitch or Weyl space to the Cantor space, cellular automata keep chaoticity properties like topological transitivity and sensitivity (but in general not equicontinuity or equicontinuous points). Vice versa, when passing from the Cantor space to one of the two others, cellular automata preserve chain transitivity and stability properties like equicontinuity, existence of equicontinuity points and stability of periodic points (but not expansivity or even sensitivity). Finally, Hedlund's theorem states that for the Cantor topology the set of all cellular automata coincides with that of continuous shift-commuting maps; this is not true in the Weyl and Besicovitch spaces: there are indeed many shift-commuting maps that are not given by any local rule.

2 Dynamical systems

A dynamical system is a continuous map $f : X \to X$ of a nonempty metric space X to itself. The n-th iteration $f^n : X \to X$ of f is defined by $f^0(x) = x$, $f^{n+1}(x) = f(f^n(x))$. A point $x \in X$ is fixed, if $f(x) = x$. It is periodic, if $f^n(x) = x$ for some $n > 0$. The least positive n with this property is called the period of x. The orbit of x is the set $\mathfrak{o}(x) = \{f^n(x) : n \geq 0\}$. A set $Y \subseteq X$ is positively invariant, if $f(Y) \subseteq Y$. A point $x \in X$ is equicontinuous ($x \in \mathcal{E}(f)$) if the family of maps f^n is equicontinuous at X, i.e. $x \in \mathcal{E}(f)$ iff

$$(\forall \varepsilon > 0)(\exists \delta > 0)(\forall y \in B_\delta(x))(\forall n > 0)(d(f^n(y), f^n(x)) < \varepsilon).$$

The map f is equicontinuous iff

$$(\forall \varepsilon > 0)(\exists \delta > 0)(\forall x \in X)(\forall y \in B_\delta(x))(\forall n > 0)(d(f^n(y), f^n(x)) < \varepsilon).$$

For an equicontinuous system $\mathcal{E}(f) = X$. Conversely if $\mathcal{E}(f) = X$ and X is compact, then f is equicontinuous; this need not be true in the non-compact case. A system (X, f) is sensitive (to initial conditions), iff

$$(\exists \varepsilon > 0)(\forall x \in X)(\forall \delta > 0)(\exists y \in B_\delta(x))(\exists n > 0)(d(f^n(y), f^n(x)) \geq \varepsilon).$$

A sensitive system has no equicontinuous point; however, there exist systems with no equicontinuity points which are not sensitive. A system (X, f) is (positively) expansive iff

$$(\exists \varepsilon > 0)(\forall x \neq y \in X)(\exists n \geq 0)(d(f^n(x), f^n(y)) \geq \varepsilon)$$

A positively expansive system on a perfect space is sensitive.

A system (X, f) is said to be (topologically) transitive if for any nonempty open sets $U, V \subseteq X$ there exists $n \geq 0$ such that $f^{-n}(U) \cap V \neq \emptyset$. If X is perfect and if the system has a dense orbit, then it is transitive. Conversely, if (X, f) is topologically transitive and if X is compact, then (X, f) has a dense orbit. Indeed, the set $\{x \in X : \overline{\mathfrak{o}(x)} = X\}$ is residual in this case. An ε-chain (from x_0 to x_n) is a sequence of points $x_0, \ldots, x_n \in X$ such that $d(f(x_i), x_{i+1}) < \varepsilon$ for $0 \leq i < n$. A system (X, f) is chain transitive if for any $\varepsilon > 0$ and any $x, y \in X$ there exists an ε-chain from x to y.

A fixed point $x \in X$ is stable if it is equicontinuous and there exists a neighbourhood $U \ni x$ such that for every $y \in U$, $\lim_{n \to \infty} f^n(y) = x$. A periodic point x with period n is stable if it is stable for f^n.

3 Cantor, Weyl and Besicovitch spaces

Let A be a finite alphabet with at least two letters. The binary alphabet is denoted by $\mathbf{2} = \{0, 1\}$. For $n \in \mathbb{N}$, denote by A^n the set of words over A of length n, by $A^* = \cup_{n \geq 0} A^n$ the set of finite words over A. We also consider words $u \in A^{[j,k]}$ indexed by an interval of integers $[j, k]$. Denote by $A^{\mathbb{Z}}$ the set of biinfinite sequences of letters of A. The i-th coordinate of a point $x \in A^{\mathbb{Z}}$ is denoted by x_i, and $x_{[j,k]} = x_j \ldots x_k \in A^{[j,k]}$ is the segment of x between indices j and k. For $u \in A^{[j,k]}$, u^∞ is the infinite repetition of u, i.e. $(u^\infty)_{m+n(k-j+1)} = u_m$ for $n \in \mathbb{Z}$ and $m \in [j, k]$. The cylinder of $u \in A^{[j,k]}$ is the set

$$[u] = \{x \in A^{\mathbb{Z}} : x_{[j,k]} = u\}.$$

The Cantor metric on $A^{\mathbb{Z}}$ is defined by

$$d_C(x, y) = 2^{-k} \quad \text{where} \quad k = \min\{|i| : x_i \neq y_i\}$$

so $d_C(x, y) < 2^{-k}$ iff $x_{[-k,k]} = y_{[-k,k]}$. The cylinders are clopen sets for d_C. It is well known that all Cantor spaces (with different alphabets) are homeomorphic. The Cantor space is compact, totally disconnected and perfect.

The Weyl pseudometric on $A^{\mathbb{Z}}$ is given by the formula

$$d_W(x, y) = \limsup_{l \to \infty} \max_{k \in \mathbb{Z}} \frac{\#\{j \in [k+1, k+l] : x_j \neq y_j\}}{l}$$

Here # means the number of elements of a set, so $d_W(x,y) < \varepsilon$ if and only if

$$(\exists l_0 \in \mathbb{N})(\forall l \geq l_0)(\forall k \in \mathbb{Z})(\#\{j \in [k+1, k+l] : x_j \neq y_j\} < l\varepsilon).$$

For $x \in A^{\mathbb{Z}}$ denote by $\widetilde{x} = \{y \in A^{\mathbb{Z}} : d_W(y,x) = 0\}$ and denote by $X_W = \{\widetilde{x} : x \in A^{\mathbb{Z}}\}$ the Weyl space over the alphabet A. Clearly every two Weyl spaces (with different alphabets) are homeomorphic. The Weyl pseudometric could be considered also on the set $A^{\mathbb{N}}$ of unilateral sequences. For $x, y \in A^{\mathbb{N}}$ put

$$d_W(x,y) \quad = \quad \limsup_{l \to \infty} \max_{k \in \mathbb{N}} \frac{\#\{i \in [k+1, k+l] : x_i \neq y_i\}}{l}$$

The map $\varphi : A^{\mathbb{Z}} \to A^{\mathbb{N}}$ defined by $\varphi(x) = x_0 x_{-1} x_1 x_{-2} x_2 \ldots$ is a homeomorphism between the unilateral and bilateral Weyl spaces. In fact φ is uniformly continuous, so it preserves completeness.

The Besicovitch pseudometric on $A^{\mathbb{Z}}$ is defined as follows:

$$d_B(x,y) = \limsup_{l \to \infty} \frac{\#\{j \in [-l, l] : x_j \neq y_j\}}{2l+1}$$

so $d_B(x,y) < \varepsilon$ if and only if

$$(\exists l_0)(\forall l \geq l_0)(\#\{j \in [-l, l] : x_j \neq y_j\} < (2l+1)\varepsilon).$$

For $x \in A^{\mathbb{Z}}$ put again $\widetilde{x} = \{y \in A^{\mathbb{Z}} : d_B(y,x) = 0\}$ and $X_B = \{\widetilde{x} : x \in A^{\mathbb{Z}}\}$ the Besicovitch space over the alphabet A; the notation is the same for the Weyl and Besicovitch equivalence classes but they will always be easy to distinguish according to context. Clearly any two Besicovitch spaces (with different alphabets) are homeomorphic and they are also homeomorphic to the unilateral Besicovitch space obtained from the pseudometric

$$d_B(x,y) \quad = \quad \limsup_{l \to \infty} \frac{\#\{i \in [0, l-1] : x_i \neq y_i\}}{l}, \quad x, y \in A^{\mathbb{N}}$$

Since $d_B(x,y) \leq d_W(x,y)$, the identity on $A^{\mathbb{Z}}$ resolves into a continuous map $I : X_W \to X_B$.

Both pseudometrics are shift-invariant: for instance $d_W(\sigma x, \sigma y) = d_W(x,y)$. In other words σ, considered as a continuous transformation on the Weyl or Besicovitch space, is an isometry.

Both the Weyl and Besicovitch spaces are homogenous. For any $u \in \mathbf{2}^{\mathbb{Z}}$, the map $f : \mathbf{2}^{\mathbb{Z}} \to \mathbf{2}^{\mathbb{Z}}$ defined by $f(x)_i = x_i + u_i \bmod 2$ is a homeomorphism, which sends 0^∞ to u. Using Toeplitz sequences, Downarowicz and Iwanik [6] show that the Weyl space is pathwise connected. In the same way we show this also for the Besicovitch space and we show also that both spaces are infinite dimensional.

A sequence $x \in A^{\mathbb{N}}$ is Toeplitz if each of its subwords occurs periodically, i.e, if

$$(\forall n \in \mathbb{N})(\exists p > 0)(\forall j \in \mathbb{N})(x_{n+jp} = x_n).$$

Toeplitz sequences are constructed by filling in periodic parts successively. For an alphabet A put $\widetilde{A} = A \cup \{*\}$. For $x, y \in \widetilde{A}^{\mathbb{N}}$, $T(x,y) \in \widetilde{A}^{\mathbb{Z}}$ is the point obtained by replacing the stars in x by y. Let t_i be the increasing sequence of all integers for which $x_{t_i} = *$. Then put

$$\begin{aligned} T(x,y)_i &= x_i \quad \text{if} \quad x_i \neq * \\ T(x,y)_{t_i} &= y_i \quad \text{otherwise} \end{aligned}$$

Consider a map $f : \{0,1\}^* \to \widetilde{A}^{\mathbb{Z}}$ defined by induction: $f(\lambda) = *^{\infty}$, then

$$\begin{aligned} f(x_0 \dots x_{n+1}) &= T(f(x_0 \dots x_n), (0*)^{\infty}) \quad \text{if} \quad x_n = 0 \\ f(x_0 \dots x_{n+1}) &= T(f(x_0 \dots x_n), (*1)^{\infty}) \quad \text{if} \quad x_n = 1. \end{aligned}$$

Thus

$$\begin{aligned} f(0) &= 0*0*0*0*0*0*\dots \\ f(1) &= *1*1*1*1*1*1\dots \\ f(00) &= 000*000*000*\dots \\ f(01) &= 0*010*010*01\dots \\ f(10) &= 01*101*101*1\dots \\ f(11) &= *111*111*111\dots \end{aligned}$$

For a real number $x \in [0,1]$ with binary expansion $x = \sum_{i=1}^{\infty} x_i 2^{-i}$ put $f(x) = \lim_{n\to\infty} f(x_1 \dots x_n)$. If $2^n x$ is never an integer for $n \in \mathbb{N}$, then x has a unique expansion and $f(x) \in \{0,1\}^{\mathbb{Z}}$; if $2^n x$ is an integer for some n, then x has two binary expansions, and $f(x)$ is the same for both expansions. It contains exactly one star, which can be filled in so that $f(x)$ is periodic. If $|x - y| < 2^{-m}$, then $x_{[1,m]} = y_{[1,m]}$; therefore $d_W(x,y) < 2^{-m+1}$ and $f : [0,1] \to X_W$ is continuous.

Proposition 1 *The Weyl and Besicovitch spaces are pathwise connected and infinite dimensional.*

Proof: Consider the continuous map $f : [0,1] \to X_W$ constructed above. Given $u \in \mathbf{2}^{\mathbb{Z}}$ the map $g : [0,1] \to \mathbf{2}^{\mathbb{Z}}$ defined by $g(x)_i = u_i f(x)_i$ is continuous, so X_W and therefore also X_B are pathwise connected. To show that X_W is infinite dimensional, construct for any n an embedding $g : [0,1]^n \to X_W$ of an n-dimensional cube by

$$g(x_1, \dots, x_n) = f(x_1)_0 \dots f(x_n)_0 f(x_1)_1 \dots f(x_n)_1 \dots \quad \square$$

The following proof is adapted from Marcinkiewicz [12].

Proposition 2 *The Besicovitch space is complete.*

Proof: We prove this for the unilateral Besicovitch space. Let $x^{(n)} \in A^{I\!N}$ be a Cauchy sequence. There exists a subsequence $x^{(n_j)}$ such that $d_B(x^{(n_{j+1})}, x^{(n_j)}) < 2^{-j-1}$. Choose a sequence l_j of positive integers such that $l_{j+1} \geq 2l_j$ and for every $l \geq l_j$

$$\#\{i \in [0,l) : x_i^{(n_{j+1})} \neq x_i^{(n_j)}\} < l \cdot 2^{-j-1}.$$

Then for $k > j$ and $l \geq l_k$

$$\#\{i \in [0,l) : x_i^{(n_k)} \neq x_i^{(n_j)}\} < l \cdot (2^{-j-1} + \cdots + 2^{-k}) < l \cdot 2^{-j}.$$

Define $x \in A^{I\!N}$ by $x_t = x_t^{(n_j)}$ if $l_j \leq t < l_{j+1}$ and x_t arbitrarily if $t < l_0$. If $k > j$ and $l_k \leq l < l_{k+1}$, then

$$\begin{aligned}
\#\{i \in [0,l) : x_i \neq x_i^{(n_j)}\} \quad &\leq \quad \#\{i \in [0,l_j) : x_i \neq x_i^{(n_j)}\} + \\
&\quad \#\{i \in [l_{j+1}, l_{j+2}) : x_i^{(n_{j+1})} \neq x_i^{(n_j)}\} + \cdots + \\
&\quad \#\{i \in [l_{k-1}, l_k) : x_i^{(n_{k-1})} \neq x_i^{(n_j)}\} + \\
&\quad \#\{i \in [l_k, l) : x_i^{(n_k)} \neq x_i^{(n_j)}\} \\
&\leq \quad l_j + (l_{j+2} + \cdots + l_k + l)2^{-j} \leq l_j + 3l \cdot 2^{-j}
\end{aligned}$$

It follows that $d_B(x, x^{(n_j)}) \leq 3 \cdot 2^{-j}$, so $x^{(n_j)}$ converges to x and since $x^{(n)}$ is a Cauchy sequence, it converges to x as well. □

To show further properties of the Weyl and Besicovitch spaces, we use Sturmian sequences (see e.g. de Luca [11] or Blanchard and Kůrka [3]). For an irrational $x \in (0,1)$ define $S(x) \in \mathbf{2}^{I\!N}$ by

$$\begin{aligned}
S(x)_n \quad &= \quad 0 \quad \text{if} \quad 0 < nx - k < 1 - x \quad \text{for some} \quad k \in I\!N; \\
S(x)_n \quad &= \quad 1 \quad \text{otherwise.}
\end{aligned}$$

$S(x)$ is called a Sturmian sequence with density x.

Lemma 1 *If $x, y \in (0,1)$ and x/y are all irrational, then*

$$d_W(S(x), S(y)) = d_B(S(x), S(y)) = x(1-y) + (1-x)y.$$

Proof: Consider the rotation

$$T(a,b) = (a + x \bmod 1, b + y \bmod 1)$$

defined on the torus $I\!R^2/\mathbb{Z}^2$; T is uniquely ergodic and its invariant measure is the Lebesgue measure. One has $S(x)_n \neq S(y)_n$ if and only if

$$T^n(0,0) \in [0, 1-x] \times [1-y, 1] \cup [1-x, 1] \times [0, 1-y];$$

this set has Lebesgue measure $x(1-y) + y(1-x)$; by unique ergodicity this is exactly the density of the set of coordinates where $S(x)_n$ and $S(y)_n$ disagree. □

Proposition 3 *The Weyl and Besicovitch spaces are neither separable nor locally compact.*

Proof: For any $0 < a < b < 1$ there exists an uncountable set $E_{ab} \subseteq (a, b)$ such that for all $x, y \in E_{ab}$, x, y and x/y are all irrationals. For every $x, y \in E_{a,b}$ one has

$$a(1-b) < d_W(S(x), S(y)) = d_B(S(x), S(y)) < b(1-a).$$

It follows that neither X_W nor X_B is separable (i.e. they do not have countable base). Since $b(1-a)$ can be arbitrarily small, and since both X_W and X_B are homogeneous, neither is locally compact. □

Let $f : A^{\mathbb{Z}} \to A^{\mathbb{Z}}$ be a W- or B-continuous map. Then $f(\widetilde{x}) \subseteq \widetilde{f(x)}$, so $\widetilde{f} : X_W \to X_W$ defined by $\widetilde{f}(\widetilde{x}) = \widetilde{f(x)}$ is continuous and $(X_W, \widetilde{f})$ (or $(X_B, \widetilde{f})$) is a dynamical system. We refer to a dynamical property of a map f that is continuous in at least one of the Cantor, Besicovitch and Weyl spaces by prefixing the letter C, B or W: for instance sensitivity in the Weyl space is called W-sensitivity. Since $d_B(x, y) \leq d_W(x, y)$ the following statements are true:

Proposition 4 *Let $f : A^{\mathbb{Z}} \to A^{\mathbb{Z}}$ be both W- and B-continuous. Then*
1. If f is W-transitive, then it is B-transitive.
2. If f is W-chain transitive, then it is B-chain transitive.
3. If f is B-expansive, then it is W-expansive.

Since the Besicovitch and Weyl spaces are not separable, no transformation can have a dense orbit. Nevertheless there exist transitive transformations on them.

Example 1 *The map f defined by $f(x)_i = x_{2i}$ is W- and B-transitive.*

This map is obviously continuous (but not shift-commuting) on both spaces. To check that it is transitive choose two points x and y, and define z by putting $z_{k.2^n} = y_k$ and $z_i = x_i$ for $i \neq k.2^n$; thus z is at distance at most 2^{-n} from x, and $f^n(z) = y$.

Proposition 5 *Let (X, f) be a dynamical system on a non-separable space X. If (X, f) is transitive, then it is sensitive.*

Proof: By Proposition 3 there exist $\varepsilon > 0$ and an uncountable set $E \subseteq X$ such that for every $x, y \in E$, $x \neq y$ one has $d(x, y) > 4\varepsilon$. We show that ε is a sensitivity constant for (X, f). Let $x \in X$. For every $n \geq 0$ there is at most one $z \in E$ whose distance from $f^n(x)$ is less than 2ε. Since E is uncountable there exists $z \in E$ such that $d(f^n(x), z) > 2\varepsilon$ for all $n \geq 0$. By transitivity, in every neighbourhood U of x there exists $y \in U$ such that $d(f^n(y), z) < \varepsilon$ for some n; hence

$$d(f^n(x), f^n(y)) \geq d(f^n(x), z) - d(z, f^n(y)) \geq 2\varepsilon - \varepsilon = \varepsilon. \quad \square$$

4 Cellular automata

A cellular automaton is a C-continuous map $f : A^{\mathbb{Z}} \to A^{\mathbb{Z}}$ that commutes with the shift $\sigma : A^{\mathbb{Z}} \to A^{\mathbb{Z}}$ defined by $\sigma(x)_i = x_{i+1}$. Every cellular automaton is defined by some local rule $F : A^{2r+1} \to A$ with radius $r \geq 0$ by

$$f(x)_i = F(x_{i-r} \dots x_{i+r}).$$

It follows that any cellular automaton is continuous for the Weyl and Besicovitch pseudometrics. From now on we compare topological and dynamical properties of cellular automata in the Cantor, Weyl and Besicovitch spaces.

Proposition 6 *A cellular automaton $f : A^{\mathbb{Z}} \to A^{\mathbb{Z}}$ is surjective if and only if it is W-surjective and if and only if it is B-surjective (that is, if $\widetilde{f} : X_W \to X_W$ or $\widetilde{f} : X_B \to X_B$ is surjective).*

Proof: Clearly if f is surjective so is $\widetilde{f}$. Suppose that $\widetilde{f} : X_W \to X_W$ is surjective. By a theorem of Hedlund [8], f is surjective if and only if every block $u \in A^*$ has a preimage. Consider the periodic point $x = u^\infty$. By the assumption there exists $y \in A^{\mathbb{Z}}$ such that $d_W(f(y), x) = 0$. It follows that in y one can find blocks that are mapped to u. The proof for d_B is identical. □

Proposition 7 *If a cellular automaton f is C-equicontinuous, then it is both W- and B-equicontinuous.*

Proof: By the assumption for $\varepsilon = 1$ there exists $\delta = 2^{-m}$ such that for every $x, y \in A^{\mathbb{Z}}$ if $x_{[-m,m]} = y_{[-m,m]}$, then $f^n(x)_0 = f^n(y)_0$ for every $n \geq 0$. Therefore, if $x_{[j-m,k+m]} = y_{[j-m,k+m]}$, then $f^n(x)_{[j,k]} = f^n(y)_{[j,k]}$ for every $n \geq 0$. Given $\varepsilon > 0$ put $\delta = \frac{\varepsilon}{2m+2}$ and suppose that $d_W(x, y) < \delta$, so there exists l_0 such that for all $l \geq l_0$ and all $k \in \mathbb{Z}$

$$\#\{i \in [k+1, k+l] : \ x_i \neq y_i\} < l\delta.$$

Thus in the interval $[k+1, k+l]$, $f^n(x)$ may differ from $f^n(y)$ only in one of the end intervals $[k+1, k+m]$, $[k+l-m+1, k+l]$ or in an interval $[i-m, i+m]$ for some i with $x_i \neq y_i$. It follows that

$$\operatorname{card}\{i \in [k+1, k+l] : \ f^n(x)_i \neq f^n(y)_i\} < l\delta(2m+1) + 2m.$$

If $l\delta > 2m$, then

$$\frac{\#\{i \in [k+1, k+l] : \ f^n(x)_i \neq f^n(y)_i\}}{l} < \delta(2m+2) = \varepsilon$$

so $d_W(f^n(x), f^n(y)) < \varepsilon$. Thus f is W-equicontinuous. The proof of B-equicontinuity is analogous. □

Proposition 8 *If a cellular automaton f has a C-equicontinuity point, then it has a W-equicontinuity point and a B-equicontinuity point.*

Proof: Let r be the radius of f and $z \in A^{\mathbb{Z}}$ be a C-equicontinuity point of f. For $\varepsilon = 2^{-r}$ there exists $\delta = 2^{-m}$ such that whenever $y_{[-m,m]} = z_{[-m,m]} = u \in A^{2m+1}$, then $f^n(y)_{[-r,r]} = f^n(z)_{[-r,r]}$ for all $n \geq 0$. We show that $x = u^\infty$ is a W-equicontinuity point. For given $\varepsilon > 0$ put $\delta = \frac{\varepsilon}{4m-2r+1}$. If $d_W(x,y) < \delta$, then there exists l_0 such that for all $l \geq l_0$ and all $k \in \mathbb{Z}$

$$\#\{i \in [k+1, k+l] : \ x_i \neq y_i\} < l\delta.$$

Every change in one of the blocks $x_{[k+1,k+2m+1]} = u$ with $k = j(2m+1)$ may change only this block or $m-r$ positions in any of its two neighbouring blocks, i.e. at most $4m-2r+1$ positions. Thus

$$\frac{\#\{i \in [k+1, k+l] : \ f^n(x)_i \neq f^n(y)_i\}}{l} < \delta(4m-2r+1) = \varepsilon$$

and $d_W(f^n(x), f^n(y)) < \varepsilon$. The proof is practically the same in the Besicovitch space. □

The following result is implicit in Hurley [9].

Lemma 2 *If $x \in A^{\mathbb{Z}}$ is a C-stable periodic point of a cellular automaton f, then $\sigma(x) = x$ and $f(x) = x$.*

Proof: Let p be the period of x. If $[u] \ni x$ is an attracting neighbourhood of x, then $\sigma(x)$ is a stable periodic point with attracting neighbourhood $\sigma([u])$. For k large enough $[u] \cap \sigma^k[u]$ and $[u] \cap \sigma^{k+1}[u]$ are both nonempty. For $y \in [u] \cap \sigma^k[u]$ and $z \in [u] \cap \sigma^{k+1}[u]$ we get

$$\sigma^k(x) = \lim_{n\to\infty} f^{np}(y) = x = \lim_{n\to\infty} f^{np}(z) = \sigma^{k+1}(x)$$

so $\sigma(x) = x$ and $x = a^\infty$. If $f(x) = b^\infty$, then $a^\infty = \lim_{n\to\infty} f^{np}(a^\infty b^\infty) = b^\infty$, so $a = b$ and $p = 1$. □

Proposition 9 *If $x \in A^{\mathbb{Z}}$ is a C-stable periodic point of a cellular automaton f, then $\widetilde{x}$ is both W-stable and B-stable.*

Proof: By Lemma 2, $x = a^\infty$ for some $a \in A$. By the proof of Proposition 8, $\widetilde{x}$ is both W- and B-equicontinuous. Since x is C-stable, there is $m > 0$ such that for $a^{2m+1} \in A^{[-m,m]}$, $\lim_{n\to\infty} f^n(y) = x$ for every $y \in [a^{2m+1}]$. Then there is s such that $f^s[a^{2m+1}] \subseteq [a^{2m+3}]$ with $a^{2m+3} \in A^{[-m-1,m+1]}$, so occurrences of a spread at least one coordinate in both directions after s steps. For the Weyl pseudometric consider a neighbourhood

$$U = \{y \in A^{\mathbb{Z}} : \ d_W(y,x) < \tfrac{1}{2m+1}\}.$$

For $y \in U$ there exists l such that for every k

$$\#\{i \in [k+1, k+l(2m+1)] : \ y_i \neq a\} < l,$$

so every subword of y of length $l(2m+1)$ contains a^{2m+1} as a subword. It follows that for $t > s(l-1)(2m+1)$, $f^t(y) = x$, so x is W-stable. For the Besicovitch pseudometric use the neighbourhood $U = \{y \in A^{\mathbb{Z}} : \ d_B(y,x) < \frac{1}{2m+1}\}$. □

Proposition 10 *If a cellular automaton f is W or B-sensitive, it is also C-sensitive.*

Proof: If f is W or B-sensitive, it has no W- or B-equicontinuity point, so by Proposition 8 it has no C-equicontinuity point and by Theorem 3 in Kůrka [10] it is C-sensitive. □

The existence of W- or B-transitive cellular automata is an open question, so that the next statement may be empty; at least it tells us where not to look for counterexamples.

Proposition 11 *If a cellular automaton f is W- or B-transitive, then it is C-transitive.*

Proof: Let f be B-transitive and $u, v \in A^{[-m,m]}$. We show that $[u] \cap f^{-n}[v] \neq \emptyset$ for some $n > 0$. Consider spatially periodic points u^∞, v^∞. By the assumption for $\varepsilon = \frac{1}{3(2m+1)}$ there exists $x \in A^{\mathbb{Z}}$ and $n > 0$ with $d_B(x, u^\infty) < \varepsilon$ and $d_B(y, v^\infty) < \varepsilon$, where $y = f^n(x)$. It follows that there is $l > 0$ such that in the interval $[-m-(2m+1)l, m+(2m+1)l]$ there are at most $(2m+1)(2l+1)\varepsilon = \frac{2l+1}{3}$ differences, i.e.

$$\begin{aligned} \#\{i \in [-m-(2m+1)l, m+(2m+1)l] : \ x_i \neq (u^\infty)_i\} &< \tfrac{2l+1}{3} \\ \#\{i \in [-m-(2m+1)l, m+(2m+1)l] : \ y_i \neq (v^\infty)_i\} &< \tfrac{2l+1}{3}. \end{aligned}$$

Thus there exists at least one unperturbed block, i.e. there is $|l_1| \leq l$ such that for $j = (2m+1)l_1$ one has

$$x_{[j-m,j+m]} = u, \ \ f^n(x)_{[j-m,j+m]} = v$$

and $\sigma^j(x) \in [u] \cap f^{-n}([v])$. For W-transitivity apply Proposition 4. □

Proposition 12 *If a cellular automaton f is C-chain transitive, then it is W- and B-chain transitive.*

Proof: Let $F : A^{[-r,r]} \to A$ be the local rule for f. A sequence $x^{(i)} \in A^{\mathbb{Z}}$ is a 2^{-m}-chain for d_C if $x_j^{(n+1)} = F(x_{j-r}^{(n)}, \ldots, x_{j+r}^{(n)})$ for $|j| \leq m$. Since only the sites $|j| \leq m+r$ are involved, we identify 2^{-m}-chains with sequences $x^{(i)}_{[-m-r,m+r]} \in$

$A^{[-m-r,m+r]}$. There exists a letter $a \in A$ such that a^∞ is periodic. Denote its period by p. Given $\varepsilon > 0$ let $m \in \mathbb{N}$ be such that $\frac{2r}{2r+2m+1} < \varepsilon$. By the assumption for every $u \in A^{[-m-r,m+r]}$ there exists a 2^{-m}-chain $u^{(1)}, \ldots, u^{(n)} \in A^{[-m-r,m+r]}$ such that $u^{(1)} = a^{2m+2r+1}$ and $u^{(n)} = u$. We can assume that $n > p$. Let $w \in A^{[-b,b]}$ be a word containing all the words $u^{(n-p+1)}, \ldots u^{(n)}$ as subwords. By the assumption again there is a 2^{-b+r}-chain from a^{2b+1} to w. Denote by q the length of this chain. If we restrict this chain to positions where $u^{(j)}$ is located, we obtain a 2^{-m}-chain of length l from $a^{2r+2m+1}$ to $u^{(j)}$. It follows that there are 2^{-m}-chains of all lengths $q, q+1, \ldots, q+p-1$ from $a^{2m+2r+1}$ to u and since a^∞ has period p there are chains from $a^{2m+2r+1}$ to u of all lengths greater than l. If we consider also chains from v to a^∞, we obtain that there exists q such that for every pair $u, v \in A^{[-m-r,m+r]}$ there exists a 2^{-m}-chain from u to v, whose length is exactly q. Given $x, y \in A^{\mathbb{Z}}$ we construct now ε-chain $x^{(1)}, \ldots, x^{(q)}$ leading from x to y for the Weyl pseudometric. In every interval

$$[b_j, c_j] = [-m - r + j(2m + 2r + 1), m + r + j(2m + 2r + 1)]$$

where $j \in \mathbb{Z}$, we construct a 2^{-m}-chain $x^{(n)}_{[b_j,c_j]}$ from $x_{[b_j,c_j]}$ to $y_{[b_j,c_j]}$, so $x^{(1)} = x$ and $x^{(q)} = y$. Moreover $f(x^{(n)})_k = x^{(n+1)}_k$ for every $k \in [b_j + m, c_j - m]$, so $x^{(n)}$ is a ε-chain for d_W. $\square$

Proposition 13 *No cellular automaton is B-positively expansive.*

Proof: Let $f : \mathbf{2}^{\mathbb{Z}} \to \mathbf{2}^{\mathbb{Z}}$ be a cellular automaton and fix $\varepsilon > 0$. Choose an integer q with $\frac{1}{q+1} < \varepsilon$ and consider points $x, y \in \mathbf{2}^{\mathbb{Z}}$ that are symmetric (i.e. $x_{-i} = x_i$ and $y_{-i} = y_i$) with nonnegative coordinates

$$\begin{aligned} x_{[0,\infty)} &= 0^{q^1} 1^{q^0} 0^{q^3} 1^{q^2} 0^{q^5} \ldots \\ y_{[0,\infty)} &= 1^{q^1} 0^{q^0} 1^{q^3} 0^{q^2} 1^{q^5} \ldots \end{aligned}$$

Then $d_B(0^\infty, x) = d_B(1^\infty, y) = \frac{1}{1+q}$. Let $F : A^{2r+1} \to A$ be the local rule of f. Now there are four possible cases; in each of them one can find a pair of points that contradicts expansivity:

1. $F(0 \ldots 0) = 0$ and $F(1 \ldots 1) = 0$; in this case $f(x) = 0^\infty$, thus for any $t \in \mathbb{N}$, $d_B(f^t(0^\infty), f^t(x)) < \varepsilon$.

2. $F(0 \ldots 0) = 0$ and $F(1 \ldots 1) = 1$; in this case $f(x) = x$, thus for any $t \in \mathbb{N}$, $d_B(f^t(0^\infty), f^t(x)) = \frac{1}{1+q} < \varepsilon$.

3. $F(0 \ldots 0) = 1$ and $F(1 \ldots 1) = 1$; in this case $f(y) = 1^\infty$, thus for any $t \in \mathbb{N}$, $d_B(f^t(1^\infty), f^t(y)) < \varepsilon$.

4. $F(0 \ldots 0) = 1$ and $F(1 \ldots 1) = 0$; in this case $f(0^\infty) = 1^\infty$, $f(1^\infty) = 0^\infty$, $f(x) = y$, $f(y) = x$, hence $\forall t \in \mathbb{N}$, $d(f^t(0^\infty), f^t(x)) = \frac{1}{1+q} < \varepsilon$. $\square$

We do not know whether the same is true in the Weyl space.
The next set of observations partly account (together with Propositions 7, 8 and 10) for the fact that passing from the Cantor to the Besicovitch and Weyl topologies considerably diminishes the set of sensitive CA.

Proposition 14 *Let f be a continuous shift-commuting map on the Weyl space. Suppose f is W-equicontinuous, or W-sensitive, or that x is a W-equicontinuity point for f: then $\sigma^n \circ f$ has the same property. The same statements are true in the Besicovitch space.*

Proof: These are immediate consequences of the facts that f commutes with the shift and that σ preserves the Weyl and Besicovitch pseudo-metrics. □

Now here are some examples showing that the converses of Propositions 7, 8, 9, 10 and 12 are false.

Example 2 *The identity map $f(x) = x$.*

The identity is W-chain transitive (since the Weyl space is connected), but not C-chain transitive (since the Cantor space is totally disconnected). Thus the converse of Proposition 12 is false.

Example 3 *The shift map $\sigma(x)_i = x_{i+1}$.*

The shift map is a W-isometry, so it is W-equicontinuous, though it is C-transitive and C-sensitive. Thus the converses of Propositions 7, 8 and 10 are not true. Observe that $\widetilde{\sigma} : X_W \to X_W$ has an infinite number of fixed points. Any sequence k_n of positive integers growing fast enough yields a fixed point

$$x = \ldots 1^{k_3} 0^{k_2} 1^{k_1} 0^{k_0} 1^{k_1} 0^{k_2} 1^{k_3} \ldots$$

Example 4 *The permutive cellular automaton $f(x)_i = x_{i-1} + x_i + x_{i+1}$*

is B-sensitive (see Cattaneo et al [4]). We do not know whether it is B-transitive.

Example 5 *The multiplication cellular automaton $f(x)_i = x_{i-1} x_i x_{i+1}$.*

The system has a C-stable fixed point 0^∞, and a W-stable and B-stable fixed point $\widetilde{0^\infty}$. In X_B and X_W $\widetilde{f}$ has many other fixed points like $0^\infty 1^\infty$, $1^\infty 0^\infty$, and when the sequence k_n grows fast enough the point

$$x = \ldots 1^{k_3} 0^{k_2} 1^{k_1} 0^{k_0} 1^{k_1} 0^{k_2} 1^{k_3} \ldots$$

.

Example 6 *Gilman's cellular automaton $f(x)_i = x_{i+1} x_{i+2}$.*

Here the fixed point 0^∞ is W-stable but not C-stable. The converse of Proposition 9 is false.

It is well known that in the Cantor topology any continuous shift-commuting map on $A^{\mathbb{Z}}$ is a cellular automaton. This is not true for the Weyl pseudometric.

Example 7 *Let the application $f : A^{\mathbb{Z}} \to A^{\mathbb{Z}}$, where $A = \{0, 1, s\}$, be defined as follows:*

$$\begin{array}{lll} f(x)_i = a+b+c & \text{if} & x_{[i-j-1,i+k+1]} = as^j bs^k c \\ f(x)_i = a+b & \text{if} & x_{[i-j-1,\infty)} = as^j bs^\infty \\ f(x)_i = b+c & \text{if} & x_{(-\infty,i+k+1]} = s^\infty bs^k c \\ f(x)_i = b & \text{if} & x_{(-\infty,\infty)} = s^\infty bs^\infty,\ x_i = b \\ f(x)_i = s & \text{if} & x_i = s \end{array}$$

where $a, b, c \in \mathbf{2}$.

The restriction of this map to $\{0,1\}^{\mathbb{Z}}$ is just the addition of the two nearest neighbours. In $A^{\mathbb{Z}}$ the occurrences of s between letters of $\{0,1\}$ play a neutral part: they stay unmodified by f but let the information pass on between occurrences of 0 and 1. By definition f commutes with the shift; a coordinate of $f(x)$ does not depend on any bounded set of neighbours, so f is not a CA. We claim it is both W- and B-continuous. First let $x \in A^{\mathbb{Z}}$ and suppose $x'_i = x_i$ except for $i = 0$; then $f(x')_i \neq f(x)_i$ for at most three values of i: 0, the first occurrence of a 0 or 1 to the left and the first one to the right. Now consider $y \in A^{\mathbb{Z}}$ and an integer $n > 0$; for each interval of coordinates $[k, k+n-1]$, $k \in \mathbb{Z}$ one has

$$\#\{j \in [k+1, k+l] :\ f(x)_j \neq f(y)_j\} \leq 3 \cdot \#\{j \in [k+1, k+l] :\ x_j \neq y_j\} + 2.$$

The first term of the right-hand sum is a very rough majoration of the differences between $f(x)$ and $f(y)$ arising in this interval from differences between x and y in the same interval; the term 2 majorates the number of differences arising in the interval because of differences between x and y outside this interval. Dividing by n and taking the limsup one obtains $d_W(f(x), f(y)) \leq 3d_W(x, y)$, and the same is obviously true for d_B, since one has only to consider one value of k for each odd n; so f is both W-continuous and B-continuous.

This example has an interesting dynamical property: there is a unique W-equicontinuous point for f. One easily shows that the fixed point $\widetilde{s^\infty}$ has this property; all other points in the Weyl space have not, because they inherit the sensitivity property of their coordinates on $\{0,1\}$.

References

[1] A.S.Besicovitch: *Almost periodic functions.* Dover 1954.

[2] F.Blanchard, P.Kůrka, A.Maass: Topological and measure-theoretical properties of one-dimensional cellular automata. *Physica D* 103 (1997) 86-99.

[3] F.Blanchard, P.Kůrka: Language complexity of rotations and Sturmian sequences. to appear in *Theoretical Computer Science*

[4] G.Cattaneo, E.Formenti, L.Margara, J.Mazoyer: A shift-invariant metric on $S^{\mathbb{Z}}$ inducing a nontrivial topology. in: Mathematical Foundations of Computer Science (I.Prívara and P.Rusika, eds.), Lecture Notes in Computer Science 1295, Springer-Verlag 1997

[5] K.Culik II, L.P.Hurd, S.Yu: Computation theoretic aspects of cellular automata. *Physica D* 45 (1990), 357-378

[6] T.Downarowicz, A.Iwanik: Quasi-uniform convergence in compact dynamical systems. *Studia Mathematica* 89:11-25, 1988

[7] A.Iwanik: Weyl almost periodic points in topological dynamics. *Colloquium Mathematicum*, 56:107-119,1988

[8] G.A.Hedlund: Endomorphisms and automorphisms of the shift dynamical system. *Math. Sys. Th.* 3 (1969), 320-375

[9] M.Hurley: Attractors in cellular automata. *Ergod. Th. & Dynam. Sys.* 10 (1990) 131-140.

[10] P.Kůrka: Languages, equicontinuity and attractors in cellular automata, *Ergod. Th. & Dynam. Syst.* (1997), 217, 417-433.

[11] A. de Luca: Sturmian words: structure, combinatorics and their arithmetics. *Theoretical Computer science*, 183: 45-82,1997.

[12] J.Marcinkiewicz: Une remarque sur les espaces de A.S. Besicovitch. *C. R. Acad. Sc. Paris* , t. 208, 1939, 157-159.

[13] S.Wolfram: *Theory and application of Cellular Automata.* World Scientific, Singapore 1986.

F.Blanchard: Institut de Mathématiques de Luminy - CNRS, Case 930, 163 avenue de Luminy, F-13288 Marseille cedex 09, France

E.Formenti: Laboratoire de l'Informatique du Parallélisme, Ecole Normale Superiéure de Lyon, 46 Allée d'Italie, F-69364 Lyon cedex 07, France.

P.Kůrka: Faculty of Mathematics and Physics, Charles University in Prague, Malostranské nám. 25, CZ-11800 Praha 1, Czechia

On Time Optimal Solutions of the Two-Dimensional Firing Squad Synchronization Problem

Kojiro Kobayashi
(Faculty of Engineering, Soka University
1-236 Tangi-cho, Hachioji-shi, Tokyo 192-0003, JAPAN
E-mail: kobayasi@t.soka.ac.jp
Fax: +81–426–91–9312)

Abstract Whether there exists a time optimal solution of the two-dimensional firing squad synchronization problem (2FSSP, for short) or not is a long standing open problem. We introduce a combinatorial problem which we call "the path extension problem for the two-dimensional array" (PEP, for short), and show that if 2FSSP has a time optimal solution then PEP has an $O(n^2)$ time algorithm. PEP seems to be a computationally hard problem and hence our result suggests that 2FSSP has no time optimal solution.

1 Introduction

The firing squad synchronization problem (FSSP, for short) is a problem in automata theory that has a long history. In this paper we consider one of its variations, the two-dimensional firing squad synchronization problem (2FSSP, for short).

One of the most fundamental problems concerning 2FSSP is whether it has a time optimal solution or not, and the problem remains still open. This paper is an attempt to attack this open problem by a complexity theoretical approach. We introduce a problem which we call "the path extension problem for the two-dimensional array" (PEP, for short), and show that if 2FSSP has a time optimal solution then PEP has an $O(n^2)$ time algorithm.

At present we do not know any lower bound for the time complexity of PEP. However, as we will show by an example later, the answer to PEP depends on the input in a subtle way, and it is quite probable that PEP has no time efficient algorithm such as an $O(n^2)$ time algorithm. Our result implies that 2FSSP has no time optimal solution under this probable assumption. It is easy to see that PEP is in NP. If we can show in the future that PEP is NP-complete, then our result implies the same under the more convincing assumption "P$\neq$NP".

Complexity theory and automata theory have had fruitful interactions. However these interactions have mainly concerned the complexity of solving problems in automata theory such as deciding equivalence of regular expressions. Our result gives an example that a solution of a purely complexity theoretical problem might lead to a solution of a purely automata theoretic problem.

First we explain the original FSSP. The problem was posed by J. Myhill in 1957. See [4] for the origin of the problem and [2] for a survey. The problem is to construct a

(deterministic) finite automaton A that satisfies some conditions. The automaton A has two inputs, I_1 from the left and I_2 from the right, and two outputs, O_1 to the left and O_2 to the right (Fig. 1(a)). The values of the outputs O_1, O_2 at a time t are the state of A at the time t. The state of A at a time $t+1$ is a function $f(s, r_1, r_2)$ of three values, the state s of A at time t, the value r_1 of its left input I_1 at time t, and the value r_2 of its right input I_2 at time t. We place identical copies $A_1, A_2, \ldots, A_n$ of A in a linear array, and connect their inputs and outputs as shown in Fig. 1(b). The value of the left input I_1 of A_1 and the value of the right input I_2 of A_n are always a special value #. Among the states of A are three special states Q, G, F. We call these three states Q, G,

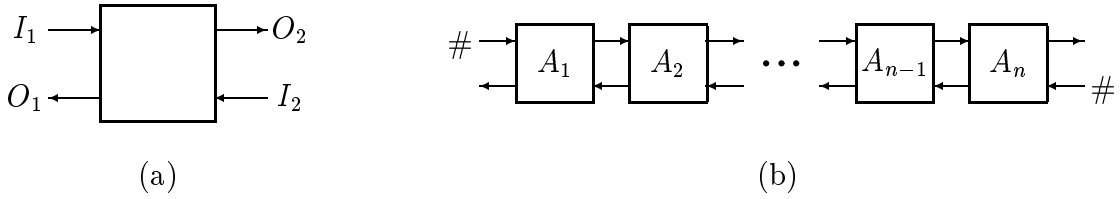


Figure 1: The original FSSP

F the *quiescent state*, the *general state* and the *firing state*, respectively. We require the value $f(\mathrm{Q}, r_1, r_2)$ to be Q for $r_1, r_2 \in \{\mathrm{Q}, \#\}$. In other words, A_i in the quiescent state cannot enter a non-quiescent state unless at least one of its neighbors A_{i-1}, A_{i+1} enters a non-quiescent state.

At time 0, the state of A_1 is G and the states of all other $A_2, \ldots, A_n$ are Q. Then the states of $A_1, \ldots, A_n$ at times 0, 1, ... are completely determined. Let $\mathrm{st}(A, n, p, t)$ denotes the state of A_p $(1 \leq p \leq n)$ at time $t(\geq 0)$, where n is the number of the copies of A in the array. The problem is to construct a finite automaton A such that, for any $n(\geq 2)$ there exists a time t_n such that

(1) $\mathrm{st}(A, n, p, t) \neq \mathrm{F}$ for any $t < t_n$ and any p $(1 \leq p \leq n)$,

(2) $\mathrm{st}(A, n, p, t_n) = \mathrm{F}$ for any p $(1 \leq p \leq n)$.

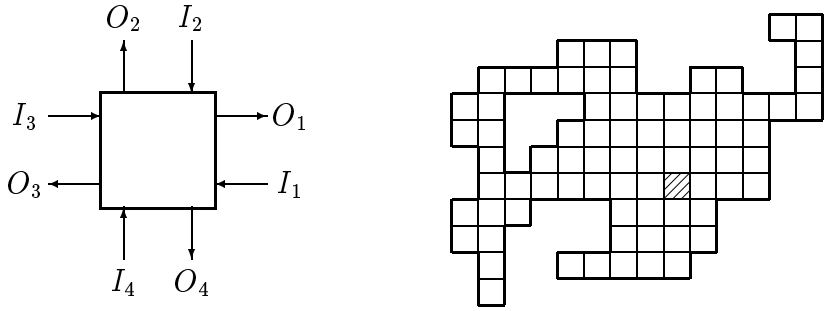
Intuitively, A_1 is a general and $A_2, \ldots, A_n$ are soldiers. At time 0 the general gives the order to fire to the soldiers, but information can be exchanged only between two adjacent soldiers at one time step. The problem is to design a rule to exchange information so that all the soldiers including the general will fire simultaneously at some time t_n. The rule must be independent of n and the information exchanged between adjacent soldiers must be from a finite set that is independent of n.

We call a finite automaton A that satisfies the above mentioned conditions a *solution* of the FSSP. For each solution A, we call the time t_n the *firing time* of the solution for an array of size n, and denote it by $\mathrm{ft}(A, n)$. To construct a solution is not difficult, and in [4] it is stated that usually it takes two to four hours for a person to find a solution.

For each n, we define the *minimum firing time* of an array of size n as $\mathrm{mft}(n) = \min\{\mathrm{ft}(A,n) \mid A \text{ is a solution}\}$, and call a solution A a *time optimal solution* if $\mathrm{ft}(A,n) = \mathrm{mft}(n)$ for any $n(\geq 2)$. We can easily show that $\mathrm{mft}(n) \geq 2n-2$ for any n (≥ 2), and hence a solution A will be time optimal if $\mathrm{ft}(A,n) = 2n-2$ for any n (≥ 2). Such a solution was first found by E. Goto in 1962. The number of the states of Goto's solution was quite large but since then the number has been reduced considerably (see [3]).

Next we explain 2FSSP. By a *position* we mean an element $p = (x,y)$ of the set $\mathbf{Z} \times \mathbf{Z}$ of all pairs of integers. We say that two positions $p = (x,y)$, $p' = (x',y')$ are *adjacent* if either $|x-x'| = 1$ and $y = y'$ or $x = x'$ and $|y-y'| = 1$. By a *configuration* we mean a pair $C = (S, p_{\mathrm{g}})$ of (i) a finite set S of positions that is connected by the above definition of adjacency, and (ii) an element p_{g} (the *general*) of S. (More precisely, we identify two configurations that are transformed to each other by the shift of the origin, and hence a configuration is an equivalence class.) Fig.2(b) shows an example of configurations. Here, each position is represented by a square and the general is marked with shadow lines.

In the variation 2FSSP, each finite automaton A has four inputs, $I_1, \ldots, I_4$, each corresponding to one of the four directions, right, up, left, down, and also four outputs $O_1, \ldots, O_4$ each corresponding to one of the four directions (Fig. 2(a)). For a configuration $C = (S, p_{\mathrm{g}})$, we place a copy of A to each position in S. The connection of inputs and outputs of copies is similar to that in the original FSSP. The value of an input that has no corresponding output is always #. At time 0, the state of the copy at the general p_{g} is G and the states of all other copies are Q.



(a) (b)

Figure 2: The variation 2FSSP

We can naturally define the state $\mathrm{st}(A,C,p,t)$ of a copy of A at a position p in a configuration C at a time t. Using this $\mathrm{st}(A,C,p,t)$ we can define "a *solution* of 2FSSP," "the *firing time* $\mathrm{ft}(A,C)$" of a solution A for a configuration C, "the *minimum firing time* $\mathrm{mft}(C)$" of a configuration C, and "a *time optimal solution* of 2FSSP." In the definition of "a *solution* of FSSP," we considered only linear arrays $A_1, \ldots, A_n$ that have at least two copies (that is, $n \geq 2$). Similarly, in the definition of "a *solution* of 2FSSP" we consider only configurations that have at least two positions. This is because of a technical reason

that many of our results fail for the special case where the configuration has only one position.

In [1] the author obtained some results on 2FSSP. We can define a variation of 2FSSP where we allow infinite state automata as solutions. Let $\mathrm{mft}_{\mathrm{inf}}(C)$ denote the value that corresponds to $\mathrm{mft}(C)$ for this variation of 2FSSP. Then the results in [1] are summarized as follows:

(1) $\mathrm{mft}(C) = \mathrm{mft}_{\mathrm{inf}}(C)$ for any C;

(2) there is a time optimal solution of this variation of 2FSSP, that is, there is an infinite state solution A such that $\mathrm{ft}(A, C) = \mathrm{mft}_{\mathrm{inf}}(C)$ for any C.

Moreover, we explicitly showed the structure of a time optimal infinite state solution A of (2), and this gave an algorithm to compute the value of $\mathrm{mft}_{\mathrm{inf}}(C)$ $(= \mathrm{mft}(C))$. However, the most interesting problem, that is, the problem whether 2FSSP has a time optimal solution or not, has remained open.

Finally we explain the path extension problem in the two-dimensional array, PEP. By a *path* we mean a nonempty finite sequence $p_1 \ldots p_n$ of different positions such that p_i and p_j are adjacent if and only if $|i - j| = 1$. We call the value n the *length* of the path, and p_1 and p_n the *start position* and the *end position* respectively. Fig. 3 shows an example of paths. The start position is shown with shadow lines. When we regard a path $p_1 \ldots p_n$ as a configuration, we assume that the start position p_1 is the general. In

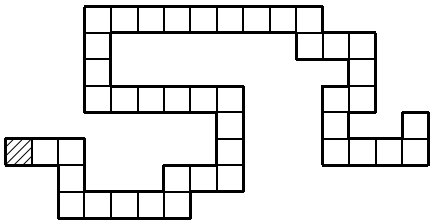


Figure 3: Path Extension Problem (PEP)

the problem PEP we are given a path $p_1 \ldots p_n$ of length n and we are required to decide whether there exists a path of the form $p_1 \ldots p_n q_1 \ldots q_n$ or not (that is, whether or not we can extend the path $p_1 \ldots p_n$ from its end position p_n to a path of total length $2n$).

In Fig. 4 (a), (b) we show two paths. They differ only in the parts in the circles. However, the answer to PEP is "YES" for (a) (we show how to extend the path with dotted lines) and "NO" for (b). This example shows that the answer to PEP depends on the form of the path in a quite subtle way. At present, the exhaustive search with backtracking is the only algorithm we know for PEP. The answers to the paths of Fig. 4 (a), (b) were obtained with this algorithm, and the computation time was 37 minutes for Fig. 4 (a) and 27 minutes for Fig. 4 (b) by a 300MHz Pentium II personal computer.

In the remainder of the paper we show that if 2FSSP has a time optimal solution then PEP has an $O(n^2)$ time algorithm. The machine model we are assuming is the standard

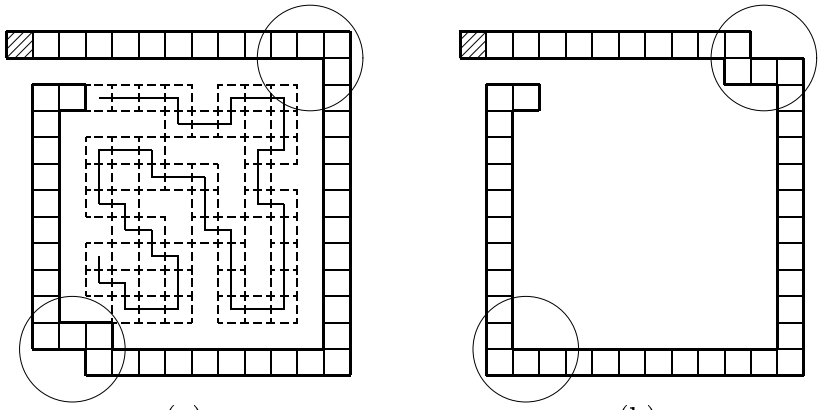


Figure 4: Examples of PEP

deterministic multitape Turing machines and the language to be decided is the set L_{PEP} of sequences $x_1x_2\ldots x_n$ of four symbols R, U, L, D such that (1) the sequence of length $n+1$ of the positions traveled by proceeding to the directions $x_1, x_2, \ldots, x_n$ starting from a position is a path, and (2) the answer to PEP for the path is "YES." Here the four symbols R, U, L, D mean to proceed to the right, up, to the left, and down, respectively.

2 Minimum firing time of paths

In this section we show an explicit formula for the minimum firing time $\mathrm{mft}(p_1\ldots p_n)$ of configurations of the form of paths $p_1\ldots p_n$.

By the *boundary condition* of a position p in a configuration C we mean the information of whether the adjacent position p' of p is in the configuration C or not for each of the four directions.

Let $p_1\ldots p_n$ be a path and i be a value such that $1 \leq i \leq n$. By $e(p_1\ldots p_n, i)$ we denote the maximum value of m such that there exists a path of the form $p_1\ldots p_iq_1\ldots q_m$ such that the boundary condition of p_i in $p_1\ldots p_n$ and that in $p_1\ldots p_iq_1\ldots q_m$ are the same. If there is no upper bound for such m then we define $e(p_1\ldots p_n, i)$ to be ∞. Equivalently, we may define $e(p_1\ldots p_n, i)$ to be the maximum value of m such that there exists a path of the form $p_1\ldots p_ip_{i+1}q_2\ldots q_m$ for $1 \leq i \leq n-1$, and to be 0 for $i = n$. We have $e(p_1\ldots p_n, 1) = \infty$, $p(p_1\ldots p_n, n) = 0$, and $e(p_1\ldots p_n, i)$ is a non-increasing function of i.

For a path $p_1\ldots p_n$ let i_0 be the value defined by

$$i_0 = \min\{i \mid 1 \leq i \leq n, i \geq e(p_1\ldots p_n, i)\}.$$

This value is well-defined because the left-hand side value i of the inequality increases from 1 to n and the right-hand side value $e(p_1\ldots p_n, i)$ decreases from ∞ to 0. We call p_{i_0} the *critical position* of the path $p_1\ldots p_n$. The critical position completely determines the minimum firing time $\mathrm{mft}(p_1\ldots p_n)$ of $p_1\ldots p_n$.

Theorem 1 *Let $p_1 \dots p_n$ be a path ($n \geq 2$) and p_{i_0} be its critical position. Then*

$$\mathrm{mft}(p_1 \dots p_n) = \begin{cases} 2i_0 - 1 & \text{if } i_0 = e(p_1 \dots p_n, i_0), \\ 2i_0 - 2 & \text{if } i_0 - 1 \geq e(p_1 \dots p_n, i_0). \end{cases}$$

(Proof) We show the proof only for the case $i_0 = e(p_1 \dots p_n, i_0)$. The proof for the case $i_0 - 1 \geq e(p_1 \dots p_n, i_0)$ is similar.

First we show $\mathrm{mft}(p_1 \dots p_n) \geq 2i_0 - 1$. Let A be an arbitrary solution of 2FSSP. If $i_0 = n$ then we have a contradiction $1 \leq i_0 = e(p_1 \dots p_n, i_0) = e(p_1 \dots p_n, n) = 0$. Hence we have $i_0 \leq n - 1$, and there are positions $q_2, \dots, q_{i_0}$ such that $p_1 \dots p_{i_0} p_{i_0+1} q_2 \dots q_{i_0}$ is a path. Let X, Y denote the paths $p_1 \dots p_n$, $p_1 \dots p_{i_0} p_{i_0+1} q_2 \dots q_{i_0}$ respectively. Then we can show the following by the mathematical induction on t:

(1) For $0 \leq t \leq i_0 - 1$,
$$\mathrm{st}(A, X, i, t) = \mathrm{st}(A, Y, i, t) \text{ for } 1 \leq i \leq t + 1,$$
$$\mathrm{st}(A, X, i, t) = \mathrm{Q} \text{ for } t + 2 \leq i \leq n,$$
$$\mathrm{st}(A, Y, i, t) = \mathrm{Q} \text{ for } t + 2 \leq i \leq 2i_0.$$

(2) For $i_0 \leq t \leq 2i_0 - 2$,
$$\mathrm{st}(A, X, i, t) = \mathrm{st}(A, Y, i, t) \text{ for } 1 \leq i \leq 2i_0 - t,$$
$$\mathrm{st}(A, Y, i, t) = \mathrm{Q} \text{ for } t + 2 \leq i \leq 2i_0.$$

Especially we have $\mathrm{st}(A, X, 1, t) = \mathrm{st}(A, Y, 1, t)$ and $\mathrm{st}(A, Y, 2i_0, t) = \mathrm{Q}$ for any $t \leq 2i_0 - 2$. From $\mathrm{st}(A, Y, 2i_0, t) = \mathrm{Q}$ and the fact that A is a solution we have $\mathrm{st}(A, Y, 1, t) \neq \mathrm{F}$. From this and $\mathrm{st}(A, X, 1, t) = \mathrm{st}(A, Y, 1, t)$ we have $\mathrm{st}(A, X, 1, t) \neq \mathrm{F}$. Hence, we have $\mathrm{st}(A, p_1 \dots p_n, 1, t) \neq \mathrm{F}$ for any $t \leq 2i_0 - 2$, and hence $\mathrm{ft}(A, p_1 \dots p_n) \geq 2i_0 - 1$. The automation A was an arbitrary solution. Hence we have $\mathrm{mft}(p_1 \dots p_n) \geq 2i_0 - 1$.

Next we show $\mathrm{mft}(p_1 \dots p_n) \leq 2i_0 - 1$.

For two positions p, q in a configuration C, the *distance* between p, q in C is "the length of the shortest path connecting p and q in C" minus one. This is the shortest time for a signal to travel from p to q in C.

Let $\mathcal{D}$ be the class of all configurations C satisfying the following conditions.

(1) The path $p_1 \dots p_{i_0}$ is included in C and p_1 is the general.

(2) For each i such that $1 \leq i \leq i_0$, the boundary condition of p_i in C is the same as its boundary condition in $p_1 \dots p_n$.

Note that the path $p_1 \dots p_n$ itself is in the class $\mathcal{D}$.

If C is a configuration in $\mathcal{D}$, then the distance between p_{i_0} and any position in C is at most i_0. This is obviously true for positions in $p_1 \dots p_{i_0}$ because the distance between p_{i_0} and p_i ($1 \leq i \leq i_0$) is $i_0 - i \leq i_0 - 1 < i_0$. Positions on the other side of p_{i_0} in C might not constitute a path. However, if there is a position p in this part of C such that the distance between p_{i_0} and p in C is at least $i_0 + 1$, this means that there is a path of the form

$p_{i_0+1}q_2 \dots q_{i_0+1}$ in C that does not touch $p_1 \dots p_{i_0-1}$, and hence $e(p_1 \dots p_n, i_0) \geq i_0 + 1$. This contradicts our assumption $i_0 = e(p_1 \dots p_n, i_0)$.

We construct a finite automaton A'. Although A' is not a solution of 2FSSP, later it will be used to construct a solution A such that $\mathrm{ft}(A, p_1 \dots p_n) \leq 2i_0 - 1$. Suppose that copies of A' are placed in positions of a configuration C. We construct A' so that A' satisfies the following conditions.

(3) If C is not in the class $\mathcal{D}$ then no copies of A' in C enter the state F.

(4) If C is in the class $\mathcal{D}$ then all copies of A' in C enter the state F at time $2i_0 - 1$.

Especially, all copies of A' in C enter the state F at time $2i_0 - 1$ if the configuration C is the path $p_1 \dots p_n$ itself. We explain the structure of A' by explaining how signals are generated, propagate and vanish in copies of A' in the configuration C.

At time 0 a signal U is generated at the position of the general. Its purpose is to see whether the configuration C is in the class $\mathcal{D}$ or not. To see this, the signal U travels from p_1 to p_{i_0} along the path $p_1 \dots p_{i_0}$ in C supposing that the position of the general of C is p_1.

If the signal U finds some i $(1 \leq i \leq i_0)$ such that either (i) the position p_i is not in C or (ii) the position p_i is in C but its boundary condition in C is different from that in $p_1 \dots p_n$, then the signal U has found that C is not in $\mathcal{D}$. In this case the signal Uvanishes instantly and no copies of A' in C enter the state F. Thus, A' satisfies the condition (3).

Suppose that the signal U has verified that for each i $(1 \leq i \leq i_0)$ there is a position p_i in the configuration C and its boundary condition in C is the same as that in $p_1 \dots p_n$. Then the signal U has found that C is in $\mathcal{D}$. In this case the signal U arrives at p_{i_0} at time $i_0 - 1$. As soon as U arrives at p_{i_0}, a new signal V_{i_0} is generated at p_{i_0} at time $i_0 - 1$, and then the signal V_{i_0} vanishes at the next time i_0. Moreover, when a signal V_i is generated at a position p in C at a time t $(1 \leq i \leq i_0)$, a signal V_{i-1} is generated at each of the adjacent positions of p in C at the time $t + 1$, and then the signal V_{i-1} vanishes at the time $t + 2$. Intuitively a signal V_i is an order to "fire after i step time." Finally, when a signal V_i is generated at a position in C at a time t $(0 \leq i \leq i_0)$, the copy of A' at the position counts i step time and enters the state F at time $t + i$.

The signal V_{i_0} is generated at p_{i_0} at time $i_0 - 1$, and the distance between p_{i_0} and any position in C is at most i_0. Hence at any position p in C a signal V_{i_0-i} is generated at time $(i_0 - 1) + i$ for some i $(0 \leq i \leq i_0)$, and the copy of A' at the position enters the state F at the time $(i_0 - 1) + i + (i_0 - i) = 2i_0 - 1$. Therefore, A' satisfies the condition (4).

It is not difficult to see that a finite number of states is sufficient to simulate this behavior of A'. Hence, we can design A' as a finite automaton. (Of course A' essentially depends on $p_1 \dots p_n$.) Let A'' be an arbitrary solution of 2FSSP and let A be the finite automaton that simulates both of A', A'' and enters F if at least one of A', A'' enters F. Then A is a solution of 2FSSP such that $\mathrm{ft}(A, p_1 \dots p_n) \leq 2i_0 - 1$. This shows that

$\mathrm{mft}(p_1 \ldots p_n) \leq 2i_0 - 1$. □

In Fig. 5 we show three examples of the calculation of the value $\mathrm{mft}(p_1 \ldots p_n)$. Let s denote the value such that $e(p_1 \ldots p_n, s-1) = \infty$, $e(p_1 \ldots p_n, s) < \infty$. Fig. 5 (a) is an example such that $s = i_0 = n$, Fig. 5 (b) is an example such that $s = i_0 \leq n-1$, and Fig. 5 (c) is an example such that $s < i_0$.

Corollary 1 *Suppose that $p_1 \ldots p_n r$ is a path ($n \geq 1$). Then $\mathrm{mft}(p_1 \ldots p_n r) \geq 2n-1$ or $\mathrm{mft}(p_1 \ldots p_n r) \leq 2n-2$ according as there exists a path of the form $p_1 \ldots p_n r q_2 \ldots q_n$ or not.*

(Proof) Suppose that there is a path of the form $p_1 \ldots p_n r q_2 \ldots q_n$, and hence $e(p_1 \ldots p_n r, n) \geq n$. We have $e(p_1 \ldots p_n r, n-1) \geq e(p_1 \ldots p_n r, n) + 1 \geq n+1$. Hence either (1) $i_0 = n$, $e(p_1 \ldots p_n r, i_0) = i_0$, or (2) $i_0 = n+1$, where i_0 is the value $\min\{i \mid 1 \leq i \leq n+1, i \geq e(p_1 \ldots p_n r, i)\}$. In the case (1) we have $\mathrm{mft}(p_1 \ldots p_n r) = 2i_0 - 1 = 2n-1$, and in the case (2) we have $\mathrm{mft}(p_1 \ldots p_n r) \geq 2i_0 - 2 = 2n$. In any case we have $\mathrm{mft}(p_1 \ldots p_n r) \geq 2n-1$.

Suppose that there is no path of the form $p_1 \ldots p_n r q_2 \ldots q_n$, and hence $e(p_1 \ldots p_n r, n) \leq n-1$. Then either (1) $i_0 = n$, $e(p_1 \ldots p_n r, i_0) \leq i_0 - 1$, or (2) $i_0 \leq n-1$. In the case (1) we have $\mathrm{mft}(p_1 \ldots p_n r) = 2i_0 - 2 = 2n-2$, and in the case (2) we have $\mathrm{mft}(p_1 \ldots p_n r) \leq 2i_0 - 1 \leq 2n-3$. In any case we have $\mathrm{mft}(p_1 \ldots p_n r) \leq 2n-2$. □

3 Path extension problem for the two-dimensional array

Now we are ready to state our main result on the relation between 2FSSP and PEP.

Theorem 2 *The answer to PEP for a path $p_1 \ldots p_n$ is "YES" if and only if there exists a position r such that $p_1 \ldots p_n r$ is a path and $\mathrm{mft}(p_1 \ldots p_n r) \geq 2n-1$.*

(Proof) We have

The answer is "YES"

$\iff \exists r[p_1 \ldots p_n r$ is a path $\wedge\ \exists q_2 \ldots q_n\ [p_1 \ldots p_n r q_2 \ldots q_n$ is a path$]]$

$\iff \exists r[p_1 \ldots p_n r$ is a path $\wedge\ \mathrm{mft}(p_1 \ldots p_n r) \geq 2n-1]$. □

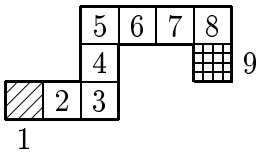
Corollary 2 *If 2FSSP has a time optimal solution then the set L_{PEP} is decidable within $O(n^2)$ time with a deterministic multitape Turing machine.*

(Proof) Assume that 2FSSP has a time optimal solution A. Suppose that we want to decide whether a sequence $x_1 \ldots x_n$ of four symbols R, U, L, D is in L_{PEP} or not. Let $p_1, \ldots, p_{n+1}$ be the sequence of positions of length $n+1$ determined by the sequence $x_1 \ldots x_n$ of symbols.

First we must test whether $p_1 \ldots p_{n+1}$ is a path or not, or equivalently whether or not there exist no i, j $(1 \leq i, i+2 \leq j \leq n+1)$ such that either $p_i = p_j$ or p_i and p_j are adjacent. This can be checked within $O(n^2)$ time.

The position where $e(p_1 \dots p_n, i)$ becomes finite for the first time (the value s).

The position where $i \geq e(p_1 \dots p_n, i)$ holds true for the first time (the value i_0).

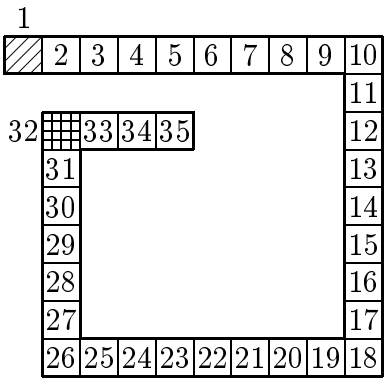


i	$e(p_1 \dots p_9, i)$
$\dots$	$\dots$
7	∞
8	∞
9	0

$s = i_0 = 9$

$\mathrm{mft}(p_1 \dots p_9) = 2i_0 - 2 = 16$

(a)

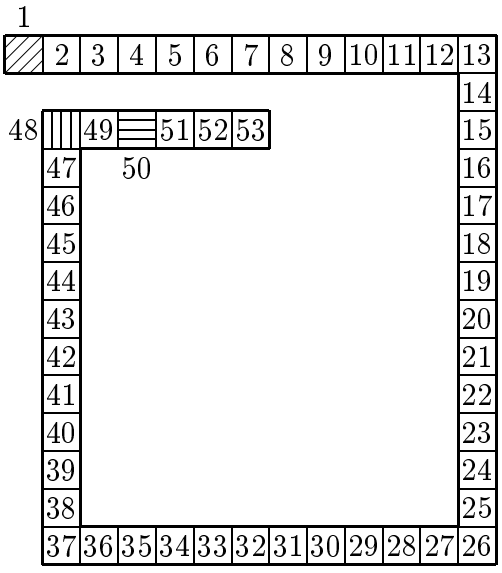


i	$e(p_1 \dots p_{35}, i)$
$\dots$	$\dots$
30	∞
31	∞
32	18
33	17
34	16
35	0

$s = i_0 = 32$

$\mathrm{mft}(p_1 \dots p_{35}) = 2i_0 - 2 = 62$

(b)



i	$e(p_1 \dots p_{53}, i)$
$\dots$	$\dots$
46	∞
47	∞
48	52
49	51
50	50
51	49
52	48
53	0

$s = 48, i_0 = 50$

$\mathrm{mft}(p_1 \dots p_{53}) = 2i_0 - 1 = 99$

(c)

Figure 5: Examples of Calculation of $\mathrm{mft}(p_1 \dots p_n)$

Suppose that $p_1 \ldots p_{n+1}$ is a path. Then it suffices to check the condition:

$$\exists r[p_1 \ldots p_{n+1} r \text{ is a path } \wedge \text{ mft}(p_1 \ldots p_{n+1} r) \geq 2n+1]$$
$$\iff \exists r[p_1 \ldots p_{n+1} r \text{ is a path } \wedge \text{ ft}(A, p_1 \ldots p_{n+1} r) \geq 2n+1].$$

There exists at most three r such that $p_1 \ldots p_{n+1} r$ is a path. To test the condition $\text{ft}(A, p_1 \ldots p_{n+1} r) \geq 2n+1$ for a specific r, we simulate the behavior of $n+2$ copies of A placed at the $n+2$ positions $p_1, \ldots, p_{n+1}, r$ until time $2n$. The simulation of one time step can be carried out within $O(n)$ time. Hence the simulation of up to time $2n$ can be carried out within $O(n^2)$ time. Hence, all the computation can be carried out within $O(n^2) + 3O(n^2) = O(n^2)$ time. □

References

[1] Kobayashi, K., On the minimal firing time of the firing squad synchronization problem for polyautomata networks, *Theoretical Computer Science* **7** (1978) 149-167.

[2] Mazoyer, J., An overview of the firing squad synchronization problem, in *Automata Networks* (C. Choffrut, ed.), LNCS 316, Springer Verlag (1986) 82-94.

[3] Mazoyer, J., On optimal solutions to the firing squad synchronization problem, *Theoretical Computer Science* **168** (1996) 367-404.

[4] Moore, E. F., The firing squad synchronization problem, in "Sequential Machines, Selected Papers" (E. F. Moore, ed.), Addison-Wesley (1964) 213-214.

Topological Mixing and Denseness of Periodic Orbits for Linear Cellular Automata over $\mathbf{Z}_m$

Luciano Margara*

Abstract

We study two dynamical properties of linear D-dimensional cellular automata over $\mathbf{Z}_m$ namely, denseness of periodic points and topological mixing. For what concerns denseness of periodic points, we complete the work initiated in [8], [2], and [1] by proving that a linear cellular automata has dense periodic points over the entire space of configurations if and only if it is surjective (as conjectured in [1]). For non-surjective linear CA we give a complete characterization of the subspace where periodic points are dense. For what concerns topological mixing, we prove that this property is equivalent to transitivity and then easily checkable. Finally, we classify linear cellular automata according to the definition of chaos given by Devaney in [7].

1 Introduction

Cellular Automata (CA) are dynamical systems consisting of a regular lattice of variables which can take a finite number of discrete values. The global state of the CA, specified by the values of all the variables at a given time, evolves according to a global transition map F based on a *local rule* f which acts on the value of each single cell in synchronous discrete time steps. A CA can be viewed as a discrete time dynamical system (X, F) where $F: X \to X$ is the CA global transition map defined over the configuration space X. CA have been widely studied in a number of disciplines (e.g., computer science, physics, mathematics, biology, chemistry) with different purposes (e.g., simulation of natural phenomena, pseudo-random number generation, image processing, analysis of universal model of computations, cryptography). For an introduction to the CA theory see [9]. CA can display a rich and complex temporal evolution whose exact determination is in general very hard, if not impossible. In particular, some properties of the temporal evolution of general CA are undecidable [4, 5, 11]. Despite their simplicity that makes it possible a detailed algebraic analysis, linear CA over $\mathbf{Z}_m$ (CA based on a linear local rule) exhibit many of the complex features of general CA. For a complete and up-to-date reference on applications of linear CA see [3].

Several important dynamical properties of linear CA, e.g., ergodicity, transitivity, sensitivity to initial conditions, and expansivity, have been studied during the last few years and in many cases exact characterizations have been obtained (see for example [10, 15, 2, 12, 14, 1]). In [13] the authors investigate and completely characterize the structure of attractors for D-dimensional linear CA over $\mathbf{Z}_m$, while in [6] the authors gives a closed formula for computing their Lyapunov exponents and their topological entropy.

*Dipartimento di Scienze dell'Informazione, Università di Bologna. Email: margara@cs.unibo.it

In this paper we study two important dynamical properties of CA: denseness of periodic points and topological mixing. We first investigate the structure of the set of periodic points of linear CA. In particular we focus our attention on a problem addressed in [1] where the authors prove that for 1-dimensional linear CA surjectivity is equivalent to have dense periodic points over the entire space of configurations leaving open the problem of characterizing this last property in the D-dimensional case. Then, we completely characterize topological mixing for linear CA in terms of the coefficients of their local rule.

The main contribution of this paper can be summarized as follows.

- We prove (Theorem 4.2) that for linear D-dimensional CA over $\mathbf{Z}_m$ ($D \geq 1,\ m \geq 2$) surjectivty is equivalent to have dense periodic points (implicitly characterizing this last property).
- For non-surjective linear D-dimensional CA over $\mathbf{Z}_m$ we explicitly characterize (Corollary 5.2) the largest subspace where periodic points are dense taking advantage of the results obtained in [13] on the attractors of linear CA over $\mathbf{Z}_m$.
- We prove (Theorem 6.2) that for linear D-dimensional CA over $\mathbf{Z}_m$ transitivity is equivalent to topological mixing (implicitly characterizing this last property).

The rest of this paper is organized as follows. In Section 2 we give basic definitions and notations. In Section 3 we prove some technical lemmas which will be useful for proving our main results. In Section 4 we prove that for linear CA surjectivity is equivalent to denseness of periodic points. In Section 5 we characterize the subspace where non-surjective linear CA have dense periodic points. In Section 6 we characterize topologically mixing linear CA. In Section 7 we provide an easy-to-check property on the coefficients of the local rule of linear CA which is equivalent to the Devaney's definition of chaos. In Section 8 we state some open problem.

2 Basic definitions

2.1 Cellular automata

For $m \geq 2$, let $\mathbf{Z}_m = \{0, 1, \ldots, m-1\}$. We consider the *space of configurations*

$$\mathcal{C}_m^D = \left\{ c \mid c \colon \mathbf{Z}^D \to \mathbf{Z}_m \right\}$$

which consists of all functions from $\mathbf{Z}^D$ into $\mathbf{Z}_m$. Each element of $\mathcal{C}_m^D$ can be visualized as an infinite D-dimensional lattice in which each cell contains an element of $\mathbf{Z}_m$. Let $s \geq 1$. A *neighborhood frame* of size s is an ordered set of distinct vectors $\vec{u}_1, \vec{u}_2, \ldots, \vec{u}_s \in \mathbf{Z}^D$. Given $f \colon \mathbf{Z}_m^s \to \mathbf{Z}_m$, a D-dimensional CA based on the *local rule* f is the pair $(\mathcal{C}_m^D, F)$, where $F \colon \mathcal{C}_m^D \to \mathcal{C}_m^D$, is the *global transition map* defined as follows.

$$[F(c)](\vec{v}) = f\left(c(\vec{v}+\vec{u}_1), \ldots, c(\vec{v}+\vec{u}_s)\right) \qquad \text{where} \qquad c \in \mathcal{C}_m^D,\ \vec{v} \in \mathbf{Z}^D. \tag{1}$$

In other words, the content of cell $\vec{v}$ in the configuration $F(c)$ is a function of the content of cells $\vec{v}+\vec{u}_1, \ldots, \vec{v}+\vec{u}_s$ in the configuration c. Note that the local rule f and the neighborhood frame completely determine F.

In order to study the topological properties of D-dimensional CA, we introduce a distance over the space of the configurations. Let $\Delta\colon \mathbf{Z}_m \times \mathbf{Z}_m \to \{0,1\}$ defined by

$$\Delta(i,j) = \begin{cases} 0, & \text{if } i = j, \\ 1, & \text{if } i \neq j. \end{cases}$$

Given $a, b \in \mathcal{C}_m^D$ the Tychonoff distance $d(a,b)$ is given by

$$d(a,b) = \sum_{\vec{v} \in \mathbf{Z}^D} \frac{\Delta(a(\vec{v}), b(\vec{v}))}{2^{\|\vec{v}\|_\infty}}, \tag{2}$$

where, as usual, $\|\vec{v}\|_\infty$ denotes the maximum of the absolute value of the components of $\vec{v}$. It is easy to verify that d is a metric on $\mathcal{C}_m^D$ and that the metric topology induced by d coincides with the product topology induced by the discrete topology of $\mathbf{Z}_m$. With this topology, $\mathcal{C}_m^D$ is a compact and totally disconnected space and F is a (uniformly) continuous map.

Throughout the paper, $F(c)$ will denote the result of the application of the map F to the configuration c, and $c(\vec{v})$ will denote the value assumed by c in $\vec{v}$. For $n \geq 0$, we recursively define $F^n(c)$ by $F^n(c) = F(F^{n-1}(c))$, where $F^0(c) = c$. Let $(\mathcal{C}_m^D, F)$ be a CA based on the local rule f. We denote by $f^{(n)}$ the local rule associated to F^n.

2.2 Linear CA over $\mathbf{Z}_m$

In the special case of linear CA the set $\mathbf{Z}_m$ is endowed with the usual sum and product operations that make it a commutative ring. In what follows we denote by $[x]_m$ the integer x taken modulo m. Linear CA have a local rule of the form $f(x_1, \ldots, x_s) = \left[\sum_{i=1}^s \lambda_i x_i\right]_m$ with $\lambda_1, \ldots, \lambda_s \in \mathbf{Z}_m$. Hence, for a linear D-dimensional CA Equation (1) becomes

$$[F(c)](\vec{v}) = \left[\sum_{i=1}^{s} \lambda_i c(\vec{v} + \vec{u}_i)\right]_m \qquad \text{where} \qquad c \in \mathcal{C}_m^D,\ \vec{v} \in \mathbf{Z}^D. \tag{3}$$

2.3 Topological properties

In this section we recall the definitions of some topological properties which determine the qualitative behavior of any general discrete time dynamical systems. Here, we assume that the space of configurations X is equipped with a distance d and that the map F is continuous on X according to the topology induced by d.

Definition 2.1 (Transitivity) *A dynamical system (X, F) is (topologically) transitive if and only if for all non empty open subsets U and V of X there exists a natural number n such that $F^n(U) \cap V \neq \emptyset$.*

Intuitively, a transitive map F has points which eventually move under iteration of F from one arbitrarily small neighborhood to any other. As a consequence, the dynamical system cannot be decomposed into two disjoint open sets which are invariant under the iterations of F.

Definition 2.2 (Topological mixing) *A dynamical system (X, F) is topologically mixing if and only if for all non empty open subsets U and V of X there exists a natural number n_0 such that for every $n \geq n_0$ we have $F^n(U) \cap V \neq \emptyset$.*

It is obvious that topological mixing implies transitivity.

Definition 2.3 (Strongly transitivity) *A dynamical system (X, F) is strongly transitive if and only if for all nonempty open set $U \subseteq X$ we have $\bigcup_{n=0}^{+\infty} F^n(U) = X$.*

A strongly transitive map F has points which eventually move under iteration of F from one arbitrarily small neighborhood to any other *point*.

Definition 2.4 (Denseness of periodic points) *Let*

$$P(F) = \{x \in X \mid \exists n \in \mathbf{N} : \ F^n(x) = x\}$$

be the set of the periodic points of F. A dynamical system (X, F) has dense periodic orbits if and only if $P(F)$ is a dense subset of X, i.e., for any $x \in X$ and $\epsilon > 0$, there exists $y \in P(F)$ such that $d(x, y) < \epsilon$.

Denseness of periodic orbits is often referred to as the *element of regularity* a chaotic dynamical system must exhibit. The popular book by Devaney [7] isolates three components as being the essential features of chaos: transitivity, sensitivity to initial conditions and denseness of periodic orbits.

3 Properties of linear CA

A D-dimensional *cylinder* $\langle(\vec{v_1}, a_1), \ldots, (\vec{v_l}, a_l)\rangle$ is a particular subset of $\mathcal{C}_m^D$ defined as

$$\langle(\vec{v_1}, a_1), \ldots, (\vec{v_l}, a_l)\rangle = \left\{x \in \mathcal{C}_m^D \mid x(\vec{v_i}) = a_i, \ i = 1, \ldots, l\right\},$$

where $a_i \in \mathbf{Z}_m$ and $\vec{v_i} \in \mathbf{Z}^D$. Note that cylinders form a basis of closed and open subsets of $\mathcal{C}_m^D$ according to the metric topology induced by the Tychonoff distance.

We first recall a result proved in [12] which holds for strongly transitive linear CA and states that for every cylinder $C \in \mathcal{C}_m^D$ it is possible to find a natural number t_C such that, with a little abuse of notation, $F^{t_C}(C) = \mathcal{C}_m^D$, i.e., every configuration of $\mathcal{C}_m^D$ can be reached after exactly t_C iterations of the map F starting from one element of C.

Lemma 3.1 ([12]) *Let F be a strongly transitive linear D-dimensional CA over $\mathcal{C}_m^D$. Then for every cylinder $C \subseteq \mathcal{C}_m^D$ there exists a natural number t_C such that*

$$\forall x \in \mathcal{C}_m^D \quad \exists c \in C : \quad F^{t_C}(c) = x.$$

□

Next lemma shows that any linear D-dimensional CA over $\mathcal{C}_{pq}^D$ with p and q relatively prime is topologically conjugated to the map $G = \left([F]_p, [F]_q\right)$ (where, for every $c \in cmd$ we define $[F]_p(c) = [F(c)]_p$.) We say that two dynamical systems (X, F) and (X', F') are *topologically conjugated* if there exists a bijective function $\psi: X \to X'$ such that $\psi(F(x)) = F'(\psi(x))$ and both ψ and ψ^{-1} are continuous (that is, ψ is a homeomorphism between X and X'). If (X, F) and (X', F') are topologically conjugated then they share the same topological properties, that is, (X, F) satisfies a given topological property if and only if (X', F') satisfies it.

Lemma 3.2 *Let F be a linear D-dimensional CA over $\mathcal{C}_m^D$ with $m = pq$ and $\gcd(p,q) = 1$. Then F is topologically conjugate to the map $G : \mathcal{C}_p^D \times \mathcal{C}_q^D \to \mathcal{C}_p^D \times \mathcal{C}_q^D$ defined by*

$$G(x_1, x_2) = \left([F]_p (x_1), [F]_q (x_2) \right) \qquad \text{where} \qquad x_1 \in \mathcal{C}_p^D,\ x_2 \in \mathcal{C}_q^D.$$

Proof. We define $\psi : \mathcal{C}_m^D \to \mathcal{C}_p^D \times \mathcal{C}_q^D$ and $\psi^{-1} : \mathcal{C}_p^D \times \mathcal{C}_q^D \to \mathcal{C}_m^D$ as follows:

$$\begin{aligned} \psi(x) &= ([x]_p, [x]_q) \\ \psi^{-1}(x_1, x_2) &= x_2 + q\,[(x_1 - x_2)\hat{q}]_p, \end{aligned}$$

where $\hat{q}$ is such that $[q\hat{q}]_p = 1$. Note that $\hat{q}$, the inverse of q modulo p, exists since p and q are relatively prime. We now prove the following properties of the map ψ:
1- $\psi^{-1}(\psi(x)) = x$ and $\psi(\psi^{-1}(x_1, x_2)) = (x_1, x_2)$,
2- ψ and ψ^{-1} are continuous, and
3- $G = \psi \circ F \circ \psi^{-1}$.
To prove property 1 we proceed as follows.

$$\begin{aligned} \psi(\psi^{-1}(x_1, x_2)) &= \left(\left[x_2 + q\,[(x_1 - x_2)\hat{q}]_p \right]_p, \left[x_2 + q\,[(x_1 - x_2)\hat{q}]_p \right]_q \right) \\ &= ([x_2]_p + [x_1]_p - [x_2]_p, [x_2]_q) = (x_1, x_2). \end{aligned}$$

To prove that $\psi^{-1}(\psi(x)) = x$ we note that for every $x \in \mathcal{C}_m^D$ there exist $k \in \mathcal{C}_q^D$ and $h \in \mathcal{C}_p^D$ such that $[x]_q + hq = x = [x]_p + kp$ and then $[x]_p = [x]_q + hq - kp$. We may write

$$\begin{aligned} \psi^{-1}(\psi(x)) &= \psi^{-1}([x]_p, [x]_q) = [x]_q + q \left[([x]_p - [x]_q)\hat{q} \right]_p \\ &= [x]_q + q\,[(hq - kp)\hat{q}]_p \\ &= [x]_q + q\,[h]_p = x. \end{aligned}$$

It is easy to verify that ψ and ψ^{-1} are continuous. To prove property 3 we proceed as follows:

$$\begin{aligned} \psi \circ F \circ \psi^{-1}(x_1, x_2) &= \psi \circ F \left(x_2 + q\,[(x_1 - x_2)\hat{q}]_p \right) \\ &= \left(\left[F \left(x_2 + q\,[(x_1 - x_2)\hat{q}]_p \right) \right]_p, \left[F \left(x_2 + q\,[(x_1 - x_2)\hat{q}]_p \right) \right]_q \right) \\ &= (y_1, y_2). \end{aligned}$$

From the linearity of F we have

$$\begin{aligned} y_1 &= \left[F(x_2) + qF([(x_1 - x_2)\hat{q}]_p) \right]_p \\ &= [F(x_2) + q\hat{q}(F(x_1) - F(x_2)]_p \\ &= [F]_p (x_1). \end{aligned}$$

Analogously, we have

$$y_2 = \left[F(x_2) + qF([(x_1, x_2)\hat{q}]_p) \right]_q = [F]_q (x_2).$$

Then $G = \psi \circ F \circ \psi^{-1}$ as claimed. □

Lemma 3.2 will be useful to prove both theorem 4.2 and theorem 6.2.

Lemma 3.3 ([6]) *Let F be a linear D-dimensional CA over $\mathcal{C}_{p^k}^D$ with local rule*

$$f(x_1, \ldots, x_s) = \left[\sum_{i=1}^{s} \lambda_i x_i \right]_{p^k},$$

and neighborhood vectors $\vec{u}_1, \ldots, \vec{u}_s$. Define

$$I = \{i \mid \gcd(\lambda_i, p) = 1\}, \qquad \hat{f} = \left[\sum_{i \in I} \lambda_i x_i \right]_{p^k},$$

and let $\hat{F}$ the global map associated to $\hat{f}$. Then, there exists $h \geq 1$ such that for all $c \in \mathcal{C}_{p^k}^D$, we have $F^h(c) = \hat{F}^h(c)$. □

Let F be a surjective linear D-dimensional CA over $\mathcal{C}_m^D$. We call F a *shift-like* CA of radius r if and only if there exist $\lambda \in \mathbf{Z}_m$ and $\vec{u} \in \mathbf{Z}^D$ with $\|\vec{u}\|_\infty = r$ such that

$$[F(c)](\vec{v}) = [\lambda\, c(\vec{v} + \vec{u})]_m \qquad \text{where} \qquad c \in \mathcal{C}_m^D,\ \vec{v} \in \mathbf{Z}^D.$$

Note that shift-like CA are surjective by definition and then from the characterization of surjective linear CA given in [10] we conclude that λ and m are relatively prime. Since $\left[\lambda^{\phi(m)}\right]_m = 1$ (where ϕ is the Euler function), we conclude that

$$[F^{\phi(m)}(c)](\vec{v}) = \lambda^{\phi(m)} c(\vec{v} + \phi(m)\vec{u}) = c(\vec{v} + \phi(m)\vec{u})$$

and then $F^{\phi(m)}$ is a *true* shift CA if $r > 0$, the identity CA if $r = 0$.

In view of the above considerations, the dynamical behavior of shift-like CA can be easily analyzed. In particular, shift-like CA with radius zero are equicontinuous and then not topologically transitive, while shift-like CA with radius greater than zero are topologically mixing and then transitive but not strongly transitive. Finally, all shift-like CA have dense periodic points.

Lemma 3.4 *Let F be a surjective but not strongly transitive linear D-dimensional CA over $\mathcal{C}_{p^k}^D$ with local rule $f(x_1, \ldots, x_s) = [\sum_{i=1}^{s} \lambda_i x_i]_{p^k}$. Then there exists a positive integer h such that the map F^h is a shift-like CA.*

Proof. Since F is surjective then there exists at least one λ_i such that $\gcd(\lambda_i, p) = 1$. Since F is not strongly transitive then for every pair λ_i, λ_j of coefficients we have that p divides at least one of them. As a consequence of the above considerations, the set $I = \{i \mid \gcd(\lambda_i, p) = 1\}$ contains exactly one element. The thesis follows from Lemma 3.3. □

4 Periodic points for surjective linear CA

In this section we prove that for linear CA over $\mathbf{Z}_m$ surjectivity implies denseness of periodic points. Since the inverse implication was already proven to be true in [1] (for general CA), we conclude that surjectivity is equivalent to denseness of periodic points.

Theorem 4.1 *Let F be a linear D-dimensional CA over $\mathcal{C}_m^D$. If F is strongly transitive then it has dense periodic points.*

Proof. It is easy to show that a CA has dense periodic points if and only if each cylinder contains at least one periodic point. We now prove that strongly transitive CA satisfy this property. Let $C \subseteq \mathcal{C}_m^D$ be any cylinder and t_C be the positive integer defined in Lemma 3.1. From Lemma 3.1 we have that there exists a sequence $c_i \in C,\ i \geq 0$, of configurations such that:

$$\forall i \geq 1: \quad F^{t_C}(c_{i+1}) = c_i.$$

Since C is compact, we can extract from c_i a convergent subsequence c_{i_j} with limit $c \in C$. We now prove that c is a periodic point of period t_C, i.e., $F^{t_C}(c) = c$. Assume by contradiction that $F^{t_C}(c) \neq c$. Then there exists a vector $\vec{v} \in \mathbf{Z}^D$ such that $[F^{t_C}(c)](\vec{v}) \neq c(\vec{v})$. Since c_{i_j} converges to c we have that

$$\exists k \in \mathbf{N} \ \ \forall j \geq k: \quad c_{i_j}(\vec{v}) = c_{i_{j+1}}(\vec{v}) = c(\vec{v})$$

which is a contradiction. Since C can be arbitrarily chosen we have that periodic points are dense. □

It remains to be proven that surjective but not strongly transitive linear CA have dense periodic points.

Theorem 4.2 *Surjective linear D-dimensional CA over $\mathcal{C}_m^D$ have dense periodic points.*

Proof. Let $m = p_1^{q_1} \cdots p_n^{q_n}$ be the prime factor decomposition of m. Let $m_i = p_i^{q_i},\ 1 \leq i \leq n$. From (a repeated application of) Lemma 3.2 we have that F has dense periodic points if and only if $[F]_{m_i}$ has dense periodic points for every $1 \leq i \leq n$. If $[F]_{m_i}$ is strongly transitive then in view of Lemma 4.1 it has dense periodic points. Assume now that $[F]_{m_i}$ is not strongly transitive. Since F is surjective, $[F]_{m_i}$ must be surjective and in view of Lemma 3.4 we conclude that there exists a positive integer h such that $[F]_{m_i}^h$ is a shift-like CA. Since shift-like CA have dense periodic points we obtain the thesis. □

5 Periodic points for non-surjective linear CA

In this section we study the distribution of periodic points of non-surjective linear CA with the aim of understanding which points of $\mathcal{C}_m^D$ can be approximated with arbitrary precision by periodic points. To this extent, we take advantage of the theory of attractors applied to linear CA developed in [13]. In [13] the authors prove that for any non-surjective linear CA F, there exists a subspace Y_F such that, for any configuration x, $F^k(x) \in Y_F$ for all $k \geq \lfloor \log_2 m \rfloor$. That is, after a transient phase of length at most $\lfloor \log_2 m \rfloor$, the evolution of the system takes place completely within the subspace Y_F. This result indicates that, in order to study periodic points of non-surjective linear CA, one should analyze the behavior of the map F over the subspace Y_F. In addition, they prove that the behavior of F over Y_F is *identical* to the behavior of a linear surjective map F^* defined over a configuration space isomorphic to Y_F.

Let F denote the global transition map of a non-surjective linear D-dimensional CA over $\mathbf{Z}_m$ defined by

$$[F(c)](\vec{v}) = \left[\sum_{i=1}^{s} \lambda_i c(\vec{v} + \vec{u}_i)\right]_m . \tag{4}$$

Let $d = \gcd(m, \lambda_1, \ldots, \lambda_s)$. Since F is not surjective we know that $d > 1$. Let $m = p_1^{q_1} p_2^{q_2} \cdots p_n^{q_n}$. Without loss of generality we can assume that $d = p_1^{v_1} p_2^{v_2} \cdots p_l^{v_l}$ with $1 \leq v_i \leq q_i$ and $l \leq n$. Let

$$q = p_1^{q_1} \cdots p_l^{q_l}, \tag{5}$$

and define

$$Y_F = \{c \in \mathcal{C}_m^D \mid [c(\vec{v})]_q = 0,\ \forall \vec{v} \in \mathbf{Z}^D\} \qquad \text{and} \qquad m^* = \frac{m}{q}. \tag{6}$$

We have the following theorem (which is the combination of Theorems 3.1 and 3.2 of [13]).

Theorem 5.1 ([13]) *Let $(\mathcal{C}_m^D, F)$ denote a non-surjective linear CA. Let Y_F and m^* be defined as in* (6)*. Then*

(a) *for any $c \in \mathcal{C}_m^D$ and $k \geq \lfloor \log_2 m \rfloor$, we have $F^k(c) \in Y_F$ and*

(b) *the subsystem (Y_F, F) is topologically conjugated to the surjective linear CA $(\mathcal{C}_{m^*}^D, [F]_{m^*})$.*

□

Taking advantage of Theorem 5.1 we can prove the main result of this section.

Corollary 5.2 *Let $(\mathcal{C}_m^D, F)$ denote a non-surjective linear CA. Let Y_F be defined as in* (6)*. Then*

(c) *the periodic points of F are dense over Y_F and*

(d) *Y_F is the largest subset of $\mathcal{C}_m^D$ where F has dense periodic points.*

Proof. From Theorem 5.1 we know that after at most $\lfloor \log_2 m \rfloor$ steps the evolution of $(\mathcal{C}_m^D, F)$ takes place completely within the subspace Y_F. This implies that all periodic points belong to Y_F. In addition, the subsystem (Y_F, F) is topologically conjugated to a surjective linear CA $(\mathcal{C}_{m^*}^D, [F]_{m^*})$ which, in view of Theorem 4.2, has dense periodic points over the entire $\mathcal{C}_{m^*}^D$. Since topological conjugation preserves denseness of periodic orbits, we conclude that F has dense periodic points over Y_F.

Let $x \in \mathcal{C}_m^D$ be any configuration which does not belong to Y_F. Then there exists a vector $\vec{v} \in \mathbf{Z}^D$ such that for every $y \in Y_F$ we have $x(\vec{v}) \neq y(\vec{v})$ and then $d(x, y) \geq 1/2^{\|v\|_\infty}$. We conclude that Y_F is the largest subset of $\mathcal{C}_m^D$ where F has dense periodic points. □

6 Topological Mixing for linear CA

In this section we prove that topological mixing and transitivity are equivalent properties as far as linear CA are concerned.

Theorem 6.1 *Let F be a linear D-dimensional CA over $\mathcal{C}_m^D$. If F is strongly transitive then it is topologically mixing.*

Proof. Let $C \subseteq \mathcal{C}_m^D$ be any cylinder and t_C be the positive integer defined in Lemma 3.1. We have that $F^{t_C}(C) = \mathcal{C}_m^D$. Since F is surjective, we have

$$\forall n \geq t_C : \quad F^n(C) = \mathcal{C}_m^D,$$

that is, F is topologically mixing as claimed. □

Theorem 6.2 *Transitive linear D-dimensional CA over $\mathcal{C}_m^D$ are topologically mixing.*

Proof. Let $m = p_1^{q_1} \cdots p_n^{q_n}$ be the prime factor decomposition of m. Let $m_i = p_i^{q_i}$, $1 \leq i \leq n$. From (a repeated application of) Lemma 3.2 we have that F is topologically mixing if and only if $[F]_{m_i}$ is topologically mixing for every $1 \leq i \leq n$. If $[F]_{m_i}$ is strongly transitive then in view of Lemma 4.1 it has dense periodic points. Assume now that $[F]_{m_i}$ is not strongly transitive. Since F is transitive, $[F]_{m_i}$ must be transitive (and then surjective). In view of Lemma 3.4 there exists a positive integer h such that $[F]_{m_i}^h$ is a shift-like CA with radius r. Since $[F]_{m_i}$ is transitive r must be greater than zero. The thesis follows from the fact that shift-like CA with radius greater than zero are topologically mixing. □

Since topologically mixing CA are transitive by definition, we conclude that for linear CA transitivity and topological mixing are equivalent properties.

7 Chaotic behavior of linear cellular automata

In this section we classify linear D-dimensional CA over $\mathbf{Z}_m$ ($D \geq 1$, $m \geq 2$) according to the Devaney's definition of chaos.

Definition 7.1 (Devaney's Chaos) *A dynamical system is chaotic according to the Devaney's definition of chaos if and only if it is topologically transitive, it is sensitive to initial conditions, and it has dense periodic points.*

Let $(\mathcal{C}_m^D, F)$ be a D-dimensional linear CA over $\mathbf{Z}_m$ defined by

$$[F(c)](\vec{v}) = \left[\sum_{i=1}^{s} \lambda_i c(\vec{v} + \vec{u}_i)\right]_m \qquad \text{where} \qquad c \in \mathcal{C}_m^D,\ \vec{v} \in \mathbf{Z}^D, \tag{7}$$

where, as usual, we assume $\|\vec{u}_1\|_\infty = 0$ and $\|\vec{u}_i\|_\infty > 0$ for every $2 \leq i \leq s$. Let $\mathcal{P}$ be the set of prime factors of m. We have the following results:

(a) $(\mathcal{C}_m^D, F)$ is topologically transitive if and only if $\gcd(\lambda_2, \ldots, \lambda s, m) = 1$ (see [2]).

(b) $(\mathcal{C}_m^D, F)$ is sensitive to initial conditions if and only if there exists $p \in \mathcal{P}$ which does not divide $\gcd(\lambda_2, \ldots, \lambda_s)$ (see [12]).

(c) $(\mathcal{C}_m^D, F)$ is surjective if and only if $\gcd(\lambda_1, \ldots, \lambda s, m) = 1$ (see [10]).

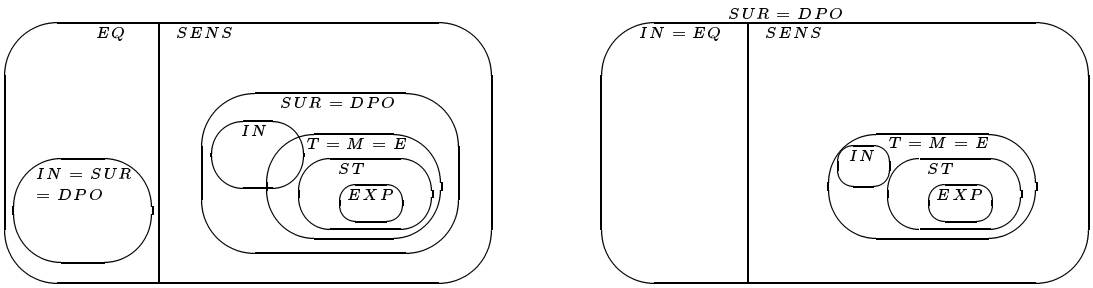
(d) $(\mathcal{C}_m^D, F)$ has dense periodic points if and only if it is surjective (this paper).

As a consequence of $(a) - (d)$ we conclude that $(\mathcal{C}_m^D, F)$ is chaotic according to the Devaney's definition of chaos if and only if it is topologically transitive, i.e.,

$$\text{Chaos according to Devaney} \quad \Longleftrightarrow \quad \gcd(\lambda_2, \ldots, \lambda s, m) = 1.$$

8 Concluding remarks and open problems

This paper extends the topological classification of linear CA over $\mathbf{Z}_m$ given in [2], [1], and [12] by exactly characterizing topological mixing and denseness of periodic orbits. In the following diagram we show inclusions among properties of D-dimensional linear CA over $\mathbf{Z}_m$ for m composite (left) and m prime (right). All inclusions are proper. Note that the class of expansive CA is empty in any dimension greater than 1.



where IN = Injectivity, T = Transitivity, ST = Strongly Transitivity, EXP = Expansivity, E = Ergodicity, M = Topological Mixing, $SENS$ = Sensitivity to Initial Conditions, SUR = Surjectivity, DPO = Denseness of Periodic Orbits, EQ = Equicontinuity.

Some of the inclusions pictured in the above diagram hold also for general (i.e., non-linear) CA. For example, expansivity implies strong transitivity and transitivity implies sensitivity. On the other hand, some of the properties which hold for linear CA do not hold for general CA. For example there exist CA which are not equicontinuous nor sensitive to initial conditions.

The following questions have a positive answer in the case of linear CA but, to our knowledge, are still unanswered in the case of general CA.
1- Transitivity implies ergodicity (with respect to the Haar measure) ?
2- Strong transitivity inmplies topologicall mixing ?
3- Surjectivity implies denseness of periodic points ?

Acknowledgements. I wish to thank G. Cattaneo and G. Manzini for many useful discussions.

References

[1] G. Cattaneo, E. Formenti, G. Manzini, and L. Margara. Ergodicity, transitivity, and regularity for linear cellular automata over Z_m. *Theoretical Computer Science*. To appear.

[2] G. Cattaneo, E. Formenti, G. Manzini, and L. Margara. On ergodic linear cellular automata over Z_m. In *14th Annual Symposium on Theoretical Aspects of Computer Science (STACS '97)*, pages 427–438. LNCS n. 1200, Springer Verlag, 1997.

[3] P. Chaudhuri, D. Chowdhury, S. Nandi, and S. Chattopadhyay. *Additive Cellular Automata Theory and Applications, Vol. 1.* IEEE Press, 1997.

[4] K. Culik, J. Pachl, and S. Yu. On the limit sets of cellular automata. *SIAM Journal of Computing*, 18:831–842, 1989.

[5] K. Culik and S. Yu. Undecidability of CA classification schemes. *Complex Systems*, 2:177–190, 1988.

[6] M. D'amico, G. Manzini, and L. Margara. On computing the entropy of cellular automata. In *25th International Colloquium on Automata Languages and Programming (ICALP '98)*. Springer Verlag, 1998.

[7] R. L. Devaney. *An Introduction to Chaotic Dynamical Systems*. Addison-Wesley, Reading, MA, USA, second edition, 1989.

[8] P. Favati, G. Lotti, and L. Margara. One dimensional additive cellular automata are chaotic according to Devaney's definition of chaos. *Theoretical Computer Science*, 174:157–170, 1997.

[9] M. Garzon. *Models of Massive Parallelism*. EATCS Texts in Theoretical Computer Science. Springer Verlag, 1995.

[10] M. Ito, N. Osato, and M. Nasu. Linear cellular automata over Z_m. *Journal of Computer and System Sciences*, 27:125–140, 1983.

[11] J. Kari. Rice's theorem for the limit set of cellular automata. *Theoretical Computer Science*, 127(2):229–254, 1994.

[12] G. Manzini and L. Margara. A complete and efficiently computable topological classification of D-dimensional linear cellular automata over Z_m. In *24th International Colloquium on Automata Languages and Programming (ICALP '97)*. LNCS n. 1256, Springer Verlag, 1997.

[13] G. Manzini and L. Margara. Attractors of D-dimensional linear cellular automata. In *15th Annual Symposium on Theoretical Aspects of Computer Science (STACS '98)*, pages 128–138. LNCS n. 1373, Springer Verlag, 1998.

[14] G. Manzini and L. Margara. Invertible linear cellular automata over Z_m: Algorithmic and dynamical aspects. *Journal of Computer and System Sciences*, 1:60–67, 1998.

[15] T. Sato. Ergodicity of linear cellular automata over Z_m. *Information Processing Letters*, 61(3):169–172, 1997.

A Geometrical Hierarchy of Graphs via Cellular Automata

Bruno Martin

LIP, ENS Lyon
46, allée d'Italie
69364 Lyon Cedex 07, France

Bruno.Martin@ens-lyon.fr

Abstract

Historically, cellular automata were defined on the lattices $\mathbb{Z}^n$, but the definition can be extended to bounded degree graphs. Given a notion of simulation between cellular automata defined on different structures (namely graphs of automata), we can deduce an order on graphs. In this paper, we link this order to graph properties and explicit the order for most of the common graphs.

Introduction

Graphs of automata are extensions of cellular automata to connected bounded degree graphs. Their formalism was first introduced by P. Rosenstiehl under the name of "intelligent graphs" [12] and studied in [9]. Graphs of automata consist in a connected bounded degree graph with a coding and decoding neighbor vector and a state on each vertex. Each vertex state uniformly evolves in discrete time step according to a definite rule involving the neighboring vertex states at previous time step ordered by the neighbor vector. The vertex states are updated simultaneously.

Simulations between cellular automata (CA) working on different architectures were studied in [10, 11]. Other notions exist (for instance [5]), but we will focus on this definition which can be very naturally extended to graphs of automata. In fact, Róka's definition may be understood as a comparison between Cayley graphs, when they are considered as graphs of automata (whatever the involved finite automaton). Then, this notion of simulation between Cayley graphs may be extended to connected bounded degree graphs. We will here study the hierarchy induced by this simulation on the set of the connected bounded degree graphs.

In [10, 11], the notion of "simple" simulation between CA defined on Cayley graphs was linked to properties of groups. Here "simple" simulation denotes unitary simulation and means that each vertex state information is indivisible and that the simulation does not need complex operations like dispatching the information of a vertex state into an infinite number of vertices of the simulating graph. We will first exhibit a more powerful sufficient condition of "simple" (unitary) simulation by linking this notion to graph properties. But direct comparison between general graphs is not easy, so we will introduce an external space and study how the graphs can be embedded in this space. This method enables to compare most of the usual graphs and to specify the hierarchy.

In a first section, we will specify the notion of graph of automata and explain the simulations we will consider. Some particular cases of simulation (elementary and unitary simulation) will be defined and studied in the second section: we will give a graph intrinsic characterization of elementary and unitary simulation. The third section describes a practical way to use this result and the consequences of these theorems on our knowledge of the hierarchy will be studied in the fourth section. In our last section, we will describe some open problems.

1 Simulation Definitions and Properties

1.1 Definitions

We will first define graphs of automata (GA), that is cellular automata defined on connected bounded degree graphs. In addition to a set of states and a transition function, we need to order the neighbors of each vertex to know how to apply the transition function.

Definition 1.1 *A* graph of automata *(or GA) on a connected bounded degree graph $G = (V, E)$ is a 5-tuple (S, G, d, N, δ) where S is a finite set called* set of states, *G is a* graph *whose degree is bounded by d, $N : V \longrightarrow (\{1, ..., d\} \longrightarrow V)$ gives for each vertex its* neighbor vector, *i.e. an order on its neighbors and $\delta : \{S \cup \epsilon\}^{d+1} \to S$ is the* transition function *where ϵ is a special element used when the vertex has less than d neighbors.*

A state is associated to each vertex of the graph. All the states change simultaneously such that the next state in $v \in V$ is given by the neighbor vector of v: $N(v)$ and the local transition δ. If v has n neighbors ($n \leq d$) $N(v)(1), ..., N(v)(n)$, the next state s' of v is given by:

$$s' = \delta(state(v), state(N(v)(1)), state(N(v)(2)), ..., state(N(v)(n)), \epsilon, ..., \epsilon)$$

This local condition naturally defines a global transition on the automaton configurations. A configuration is a function $V \longrightarrow S$ which associates to each vertex its state. Let us define $C_{\mathcal{A}} = S^V$ the set of all configurations of the automaton $\mathcal{A}$. The GA transforms a configuration c to its successor $\mathcal{A}(c)$ such that:

$$\forall c \in C_{\mathcal{A}}, \forall v \in V, \mathcal{A}(c)(v) = \delta(c(v), c(N(v)(1)), c(N(v)(2)), ..., c(N(v)(n)), \epsilon, ..., \epsilon)$$

Now define the notion of simulation used in [10, 11]. A GA $\mathcal{A}$ is simulated by $\mathcal{B}$ if we can encode the configurations of $\mathcal{A}$ into configurations of $\mathcal{B}$ such that after T steps of $\mathcal{B}$ we obtain the code of the next step of $\mathcal{A}$, formally:

Definition 1.2 *Let $\mathcal{A}$ and $\mathcal{B}$ be two graphs of automata, $\mathcal{B}$ simulates $\mathcal{A}$ if there exists an injective function $f : C_{\mathcal{A}} \longrightarrow C_{\mathcal{B}}$ from the set of configurations of $\mathcal{A}$ into the set of configurations of $\mathcal{B}$ and a constant T such that for all the configurations $c \in C_{\mathcal{A}}$*

$$f(\mathcal{A}(c)) = \mathcal{B}^T(f(c))$$

i.e. the following diagram commutes:

$$\begin{array}{ccc} c & \xrightarrow{f} & f(c) \\ \mathcal{A}\Big\Vert & & \Big\Vert\mathcal{B}^T \\ \mathcal{A}(c) & \xrightarrow{f} & \mathcal{B}^T(f(c)) \end{array}$$

Remark that there is no condition over f, for instance f can burst the information of a state into different vertices (an unidimensional GA whose states are couples can be simulated by another unidimensional GA with the function $(..., (s_0, s_1), (s_2, s_3), ...) \longmapsto (..., (s_{-1}, s_0), (s_1, s_2), ...)$). In addition, some states may appear on an infinite number of vertices of the simulating graph (see the simulation of $\mathbb{Z}^2$ by F_2 in section 4). When this does not happen (i.e. when a vertex is simulated by only one vertex of the simulating graph), we will say that the simulation is *unitary*.

Definition 1.3 *A simulation is* unitary *if and only if*

$$\forall v \in G_{\mathcal{A}}\ \exists(!)v' \in G_{\mathcal{B}}\ \ \forall c \in C_{\mathcal{A}}\ \ \forall \alpha \in S\ \ \exists \beta \in S\ \ \ f(c_{v,\alpha}) = f(c)_{v',\beta}$$

where $c_{v,\alpha}$ is the configuration c in which the state of v is replaced by α.

An unitary simulation defines an implicit function $G_{\mathcal{A}} \longrightarrow G_{\mathcal{B}}$ which associates to each vertex the only vertex of $G_{\mathcal{B}}$ which simulates it. If this function is injective, the simulation is said to be *elementary*.

From this definition, we deduce an order on the connected bounded degree graphs in the following definition:

Definition 1.4 *Let G and G' be two bounded degree graphs, we will denote $G \leq G'$ and, loosely speaking, say G' simulates G if and only if each graph of automata on G can be simulated by a graph of automata on G'. This relation on graphs will be called simulation relation or simulation order.*

In a very natural way, we define the notions of *elementary* and *unitary simulation* when all the simulations can be elementary or unitary. We will prove in the next section that these two simulations defined on graphs through graphs of automata can be expressed intrinsically in graph theory.

1.2 Necessary Condition of Simulation

Let us recall here an important theorem of [10, 11]. Given two graphs G and G', if G' simulates G, then there exists a constant C such that the graph growth of G is less than C times the graph growth of G' where the graph growth is defined as follows:

Definition 1.5 *Let G be a graph and v a vertex of G. The graph growth is the function $\mathbb{N} \longrightarrow \mathbb{N}$ which associates to n the number of vertices v' such that the distance between v and v' is less than n (i.e. such that there is a path between v and v' of length lower than n).*

Let us notice that on connected graphs the asymptotic behavior of the graph growth makes sense because it is independent of the initial vertex.

Graph growth indicates the amount of information that a vertex can access to in n steps of the graph of automata evolution. This theorem just means that if whatever the slowdown constant T and even after a grouping operation, a graph cannot access in nT steps to the information that another graph can access to in n steps, then the first graph cannot simulate the second one.

Let us specify here formally what we mean by a grouping operation. We can see a graph of automata as a graph with a finite automaton on each vertex. If it has enough states, a finite automaton can simulate a graph of automata defined on a finite graph. So given a graph of automata on G, and a finite subgraph G_0 of G, the behavior of the graph of automata on G_0 can be simulated by a finite automaton linked to all the neighbors in $G \setminus G_0$ of the vertices in G_0 without changing the behavior of the other vertices of G. We denote this a grouping operation because the finite automata on each vertex of G_0 are grouped into one finite automaton with more states. By definition, a grouping operation leads to an unitary simulation. Of course, the composition of a finite number of grouping operation remains a grouping operation. In addition, we can perform simultaneously an infinite number of grouping operation if the number of grouped vertices is bounded, this leads to the following definition.

Definition 1.6 *Let G be a connected bounded degree graph and $(G_i)_{i\in I}$ a set of finite subgraphs of G whose number of elements is bounded by M and such that, $\forall i \neq j,\ G_i \cap G_j = \emptyset$[1]. Let G' be the graph G where all the subgraphs G_i are successively replaced by vertices v_i linked to all the neighbors of the vertices of G_i in G. Let $\mathcal{A} = (S, G, d, N, \delta)$ be a graph of automata on G and $\mathcal{A}' = (S', G', d', N', \delta')$ a graph of automata on G'.*

Let us suppose that there exists $(f_i)_{i\in I}$ a family of injective functions $f_i : S^{G_i} \longrightarrow S'$ and an injection $\iota : S \longrightarrow S'$. $\mathcal{A}'$ is a grouped automaton of $\mathcal{A}$ if $\mathcal{A}$ is simulated without slowdown by $\mathcal{A}'$ with respect to the function

$$f \left| \begin{array}{l} C_{\mathcal{A}} \longrightarrow C'_{\mathcal{A}} \\ c \longmapsto c' \text{ where } \left\{ \begin{array}{l} \text{if } v \notin (v_i)_{i\in I} \text{ then } c'(v) = \iota(c(v)) \\ \text{else } c'(v_i) = f_i(c\,|_{G_i}) \end{array} \right. \end{array} \right.$$

For instance, we can simulate a graph of automata on $\mathbb{Z}$ by another one on $\mathbb{Z}$ by grouping the states two by two. This simulation is done in real time, but let us notice that the simulating automaton has enough information to compute the two next steps. This kind of regular grouping over $\mathbb{Z}$ with acceleration is studied in [7]. But here we are not interested in any acceleration but in the possibility that each vertex of a simulating graph can simulate a bounded number of vertices of the simulated one.

After this specification about grouping operation, let us give two important corollaries of Zsuzsanna Róka's theorem:

- $\mathbb{Z}^p \not\leq \mathbb{Z}^q$ if $p > q$ because $\mathbb{Z}^p$ growth behaves like n^p;
- $\forall p \quad F_2 \not\leq \mathbb{Z}^p$ where F_2 is the free group with two generators because F_2 growth is $4 \times 3^{n-1}$.

2 Elementary and Unitary Simulations

Let $\mathcal{A}$ and $\mathcal{B}$ be two graphs of automata. If the simulation of $\mathcal{A}$ by $\mathcal{B}$ is unitary, then we can naturally define the function ϕ from $G_{\mathcal{A}}$ to $G_{\mathcal{B}}$ which associates to each vertex of $G_{\mathcal{A}}$ the only vertex of $G_{\mathcal{B}}$ which simulates it. If $\mathcal{A}$ is "complicated enough", each vertex needs the states of all its neighbors to compute its next state. But $\mathcal{B}$ simulates a step of $\mathcal{A}$ in T steps, so the distance in $G_{\mathcal{B}}$ between vertices simulating two neighbors of $G_{\mathcal{A}}$ is less than T. This exactly means that T is a Lipschitz constant for ϕ because connected graphs are (discrete) metric spaces, the distance between two vertices is the length of the smallest path between them and the Lipschitz condition definition is the following:

[1] This implies that I is countable because G is countable (it is a connected bounded degree graph).

Definition 2.1 *Let $(\mathcal{E}, d_\mathcal{E})$ and $(\mathcal{F}, d_\mathcal{F})$ be two metric spaces. A function $f : \mathcal{E} \longrightarrow \mathcal{F}$ satisfies the Lipschitz condition if and only if*

$$\exists K \in \mathbb{R} \ \forall (x, y) \in \mathcal{E}^2 \quad d_\mathcal{F}(f(x), f(y)) \leq K d_\mathcal{E}(x, y)$$

We say that K is a Lipschitz constant for f.

In addition, only a finite number of vertices may be sent by ϕ on a given vertex because the simulating GA has only a finite number of states. So the existence of an injective (resp. with a bounded number of antecedents) function from $G_\mathcal{A}$ to $G_\mathcal{B}$ which satisfies the Lipschitz condition is necessary to have $G_\mathcal{A}$ simulated by $G_\mathcal{B}$ in an elementary (resp. unitary) way. In fact, this existence is also sufficient, as we will see next, so we have the two following results:

Theorem 2.1 *Let $G_\mathcal{A}$ and $G_\mathcal{B}$ be two bounded degree graphs, $G_\mathcal{B}$ is elementarily simulated by $G_\mathcal{A}$ if and only if there exists an injective function $f : G_\mathcal{A} \longrightarrow G_\mathcal{B}$ which satisfies the Lipschitz condition.*

Theorem 2.2 *Let $G_\mathcal{A}$ and $G_\mathcal{B}$ be two bounded degree graphs, $G_\mathcal{B}$ is unitarily simulated by $G_\mathcal{A}$ if and only if there exists a function $f : G_\mathcal{A} \longrightarrow G_\mathcal{B}$ with a bounded number of antecedents which satisfies the Lipschitz condition.*

To prove the sufficient parts of these theorems, the idea is to simulate each vertex by its image by the application. The images of two neighbors are separated by at most M vertices where M is the Lipschitz constant of f, so the graph of automata waits M steps to propagate the information and then the calculus is made thanks to a static information which indicates to each vertex the position of its neighbors in the simulated graph.

Proof 2.1 of both theorems 2.1 and 2.2

- Necessary part:
  Let us suppose that G' elementarily (resp. unitarily) simulates G and let us prove the existence of an injective (resp. with a bounded number of antecedents) function $f : G \longrightarrow G'$ which satisfies the Lipschitz condition. We consider the graph of automata on G with two states (0 and 1) that only propagates 1 to all its neighbors, this is the "complicated enough" GA we need. This GA is elementarily (resp. unitarily) simulated by a GA defined on G', let f be the function from G into G' which associates to each vertex v the only vertex of G' whose states depend of the state of v. f has a bounded number of antecedents because of the finite number of states of the simulating automaton. If the simulation is elementary, by definition, f is injective.

  Let T be the slowdown constant of the simulation, let us prove that T is a Lipschitz constant for f. Let x and y be two neighbor vertices of G. Let us consider the initial configuration where x contains the state 1, and all the other vertices contain the state 0. In T steps of the simulating GA, the state of $f(y)$ has been changed, so there is a path between $f(x)$ and $f(y)$ whose length is less than T.

- Sufficient part:
  For short we will prove that for injective functions f which satisfy the Lipschitz condition (theorem 2.1), there is an elementary simulation. If f has a bounded number of antecedents (theorem 2.2), the proof is exactly the same by a grouping operation (see definition 1.6), but of course, the simulation is no more elementary but unitary.

  Let f be an injective function satisfying the Lipschitz condition from $G_\mathcal{A}$ into $G_\mathcal{B}$, a graph of automata $\mathcal{A} = (S, G_\mathcal{A}, d, N, \delta)$ and the neighbor vectors N' over $G_\mathcal{B}$. Let us build $\mathcal{B} = (S', G_\mathcal{B}, d', N', \delta')$ on $G_\mathcal{B}$ which elementarily simulates $\mathcal{A}$. With word, we will send the state of the vertex v of $G_\mathcal{A}$ into the state of the vertex $f(v)$. To simulate a step of $\mathcal{A}$, if M is a Lipschitz constant for f, M steps are enough to bring the states of the neighbors vertices. To calculate the next step of $\mathcal{A}$, the local transition function of $\mathcal{B}$ just needs to know which are the neighbors on $G_\mathcal{A}$ among all the vertices in the M radius ball on $G_\mathcal{B}$. The set of all possibilities is finite, so this information can be statically encoded in each vertex.

  Hence the finite set of states of $\mathcal{B}$ is:

$$S' = ((S \cup \{\omega\}) \cup (S \cup \{\omega\})^d \cup ... \cup (S \cup \{\omega\})^{d^M}) \times (C \cup \{\epsilon\})$$

  where C represents the encoding of the neighbors in $G_\mathcal{A}$ (it is the finite set of functions $(S \cup \{\omega\})^{d^M} \rightarrow S^d$ which gives the neighbors in the simulated graph among the elements of the M radius ball in the simulating graph).

Hence the function $g : C_{\mathcal{A}} \longrightarrow C_{\mathcal{B}}$ is defined as follows: if $v \notin f(G_{\mathcal{A}})$ then the state of v is (ω, ϵ). If $v = f(v')$, v' containing the state s, then its state will be (s, c), where $c \in C$ encodes the neighbors of v' in $G_{\mathcal{A}}$ among the vertices in the M radius ball centered in v in $G_{\mathcal{B}}$.

δ' never changes the encoding part of the states. It stores during M steps the states of all its neighbors. Then, thanks to the encoding, it simulates the transition of $\mathcal{A}$: $\delta'((s, c)$, states of the neighbors vertices$) = (\delta(c(s)), c)$. So the slowdown constant is here $M + 1$, but for simplicity the last step does not depend of the neighbors states. It is possible to do the last two steps in only one, the slowdown constant is in this case M.

□

Let us notice that there exist non unitary simulations, for example the simulation of $\mathbb{Z}^2$ by the free group with two generators F_2. We will prove in our third section that $\mathbb{Z}^2 \leq F_2$ (each vertex of $\mathbb{Z}^2$ will be simulated by an infinite number of vertices of F_2), and there is no unitary simulation because of the following theorem:

Theorem 2.3 *There exists no application satisfying the Lipschitz condition with a bounded number of antecedents from $\mathbb{Z}^2$ into F_2.*

Proof 2.2 Let us suppose by refutation that a function satisfying the Lipschitz condition f with a bounded number of antecedents exists. F_2 is a tree and as $\mathbb{Z}^2$ is not simulated by $\mathbb{Z}$ (see the first section for graph growth reasons), there are two branches of F_2 with an infinite number of vertices of $f(\mathbb{Z}^2)$.

So we can separate F_2 in two trees $F_2 = T_1 \cup T_2$ (see figure 1) with an infinite number of vertices in $f(\mathbb{Z}^2)$ in each one such that T_2 is a subtree of F_2 (so $T_1 = F_2 \setminus T_2$).

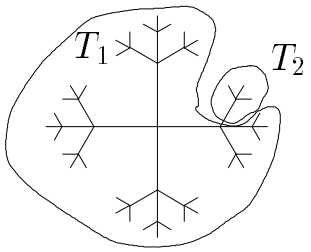


Figure 1: The partition of F_2, involved in the proof 2.2

There are an infinite number of couples (x, y) of neighbor vertices in $\mathbb{Z}^2$ such that $f(x) \in T_1$ and $f(y) \in T_2$. To prove this, let us define $B_{f^{-1}(T_2)}$, the $f^{-1}(T_2)$ border (i.e. the set of all vertices of $f^{-1}(T_2)$ having a neighbor in $f^{-1}(T_1)$). Let us suppose by refutation that $B_{f^{-1}(T_2)}$ is finite, so it is included in a ball B of F_2. The vertices out of B are all either in $f^{-1}(T_1)$ or in $f^{-1}(T_2)$ because between each couple of vertices out of B, there are a path which does not cross the border. Hence $f^{-1}(T_1)$ or $f^{-1}(T_2)$ is included in B and so is finite which is impossible.

Now, as f satisfies the Lipschitz condition, all the elements of $f(F_{f^{-1}(T_2)})$ are at a bounded distance from T_1, which is again impossible because of the bounded number of antecedents. □

3 A Practical Way to Prove Unitary and Elementary Simulations

Given two graphs, it is not easy to know whether there exists an injective function from one graph into the other one which satisfies the Lipschitz condition. But let us try to find such an application from $\mathbb{Z}^2$, the two dimensional lattice into the hexagonal tiling of the plane (see figure 2). We can associate to each vertex of the lattice, the nearest vertex of the hexagonal tiling. Clearly this application satisfies the Lipschitz condition and only a finite number of vertices may have the same image. So the lattice is unitarily simulated by the hexagonal tiling. This reasoning is true for any embedding of the lattice and the hexagonal tiling into the plane. So by similarity, we can reduce the scale of the hexagonal tiling such that the nearest point of two different vertices of the lattice are always different (like on the figure 2). So in fact, there exist elementary simulations of the lattice by the hexagonal tiling.

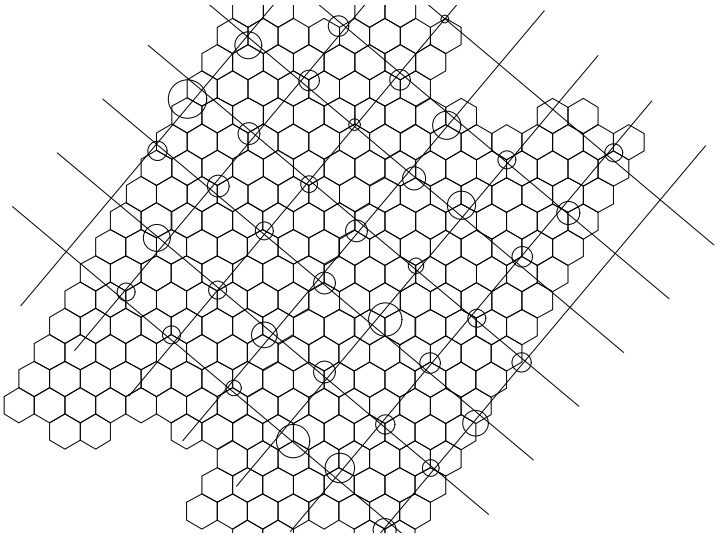


Figure 2: The lattice $\mathbb{Z}^2$ can be simulated by the hexagonal tiling by simulating each vertex of the lattice by the nearest vertex of the hexagonal tiling

In the same way, all the tilings of the plane $\mathbb{R}^2$ are equivalent by the elementary simulation, and furthermore for all n, all the tilings of $\mathbb{R}^n$ are equivalent to each other. We denote C_n the class of all the graphs equivalent to the tilings of $\mathbb{R}^n$ (including the n-dimensional lattice $\mathbb{Z}^n$). All these classes are different because $\mathbb{Z}^n \leq \mathbb{Z}^m$ iff $n \leq m$ for graph growth reasons. We will now specify and improve our method to find functions between graphs which satisfy the Lipschitz condition.

To find an injective application from the lattice to the hexagonal tiling satisfying the Lipschitz condition, we use an external space: $\mathbb{R}^2$. To be able to define the nearest vertex (or one of the nearest, because when several vertices are the nearest, we shall chose arbitrarily one of them), the external space needs to be metric and with compact closed balls. This last condition makes sure that the distance from a point to a closed set exists and is reached by a point of the set, so, given a point, we can define the (finite) set of the nearest points and chose arbitrarily one among them.

In addition, the simulated and simulating graphs do not need to be tilings of the external space. In fact, the simulated graph needs only to be embedded in this space with bounded edges and a minimum distance between two vertices (so that only a finite number of vertices may have the same nearest point in the simulating graph). Graphs satisfying these two properties will be called plungings:

Definition 3.1 *Let $\mathcal{E}$ be a metric space with compact closed balls and $G = (V, E)$ be a graph. G can be plunged into $\mathcal{E}$ if there is a function $f : V \to \mathcal{E}$ such that:*

- *$\exists \epsilon \in \mathbb{R}^{*+}$ such that $\forall x, y \in V$, $d_{\mathcal{E}}(f(x), f(y)) > \epsilon$ if $d_{\mathcal{E}}$ represents the distance of $\mathcal{E}$ (i.e. there is a minimal distance between two points of $f(V)$);*
- *f satisfies the Lipschitz condition, i.e. $\exists M \in \mathbb{R}^+$ such that for all $(x, y) \in E$, $d_{\mathcal{E}}(f(x), f(y)) \leq M$.*

Remark 3.1 *By misnomer, we will identify the vertices of the plunged graph with their image in $\mathcal{E}$ by f.*

As for the simulated graph, the simulating one must satisfy the plunging properties and in addition cover all the space (i.e. the distance from any point of the space to the embedding of the simulating graph is bounded), and the graph distance has to be equivalent to the one induced by the space distance through embedding (to ensure the application will satisfy the Lipschitz condition). Such graphs will be said covering of the space, formally,

Definition 3.2 *Let $\mathcal{E}$ be a metric space with compact closed balls and $G = (V, E)$ be a graph. G is a covering of $\mathcal{E}$ if G can be plunged into $\mathcal{E}$ and if it satisfies the two following conditions:*

- *$\exists M' \in \mathbb{R}^+$ such that for all $x \in \mathcal{E}$, $\exists v \in V$ with $d_{\mathcal{E}}(x, f(v)) \leq M'$ (i.e. the distance from a point of $\mathcal{E}$ to $f(V)$ is bounded);*
- *The distance on G induced by the distance on $\mathcal{E}$ is equivalent to the distance of the graph G, that is there exists a constant k such that for all couples of vertices x and y, there exists a path from x to y whose length is less than or equal to $k d_{\mathcal{E}}(x, y)$.*

Let us remark that all the graphs that we intuitively want to call tilings of the space satisfy the covering properties, for instance the $\mathbb{Z}^n$ lattice is a covering of $\mathbb{R}^n$.

The definitions of plunging and covering have been designed to ensure the existence of a function satisfying the Lipschitz condition with a bounded number of antecedents, so we can express the following theorem:

Theorem 3.1 *Let $\mathcal{E}$ be a metric space with compact closed balls, if G is a plunging into $\mathcal{E}$ and G' is a covering of $\mathcal{E}$, then $G \leq G'$ with a unitary simulation.*

Proof 3.1 We embed G and G' in $\mathcal{E}$ such that the embeddings satisfy the plunging and covering properties. Let g be the function from G to G' which associates to each vertex of G the nearest vertex of G' (if there are several nearest points, we chose one arbitrarily). g is well defined because $\mathcal{E}$ has compact closed balls: the embedding of G' in $\mathcal{E}$ is a closed set so the minimal distance between any point and this set is reached. As the embedding of G' is discrete, only a finite number of vertices may be at the smallest distance.

Given a point of G', the set of points of $\mathcal{E}$ which are nearer this point of G' than any other point of G' is (by compactness of closed balls) included into the compact ball of radius M' by definition of covering. As only a finite number of balls with radius $\epsilon/2$ may be put into this ball, only a finite number of vertices of G may be mapped into the same vertex of G' by g.

In addition, if l is a Lipschitz constant for the plunging of G into $\mathcal{E}$, the distance in $\mathcal{E}$ of two images by g of two neighbors in G is bounded by $l + 2M$. But the distance in G' is equivalent to the distance in $\mathcal{E}$ so the distance in G' of the image by g of two neighbors in G is bounded by $k(l + 2M)$, so g is an application from G into G' which satisfies the Lipschitz condition.

So by the last section theorem, G' unitarily simulates G. □

If similarities are possible on $\mathcal{E}$, that means practically that $\mathcal{E}$ is a normed vectorial space with compact closed balls, which is equivalent to $\mathcal{E}$ is a normed vectorial space with finite dimension, i.e. if $\mathcal{E} = \mathbb{R}^n$, then the simulations are elementary. For instance, here are some metric spaces which are not vectorial: all the connected bounded degree graphs, the sphere, the cylinder, the cone, the paraboloïd of revolution and the hyperbolic space H_2 (see below). Let us notice that all the simulations deduced from the cylinder or the cone may be proved thanks to $\mathbb{R}$ or $\mathbb{R}^2$ and can also be elementary.

Corollary 3.1 *If G is a plunging into $\mathbb{R}^n$ and G' is a covering of $\mathbb{R}^n$ then the simulation of G by G' is elementary.*

4 Consequences on the Hierarchy

4.1 Main Features of the Hierarchy

The application of the last section theorems to the vectorial spaces $\mathbb{R}^n$ allows us to define the C_n classes (containing $\mathbb{Z}^n$). In addition, the simulations we can deduce over $\mathbb{R}^n$ are elementary thanks to the similarity possibility.

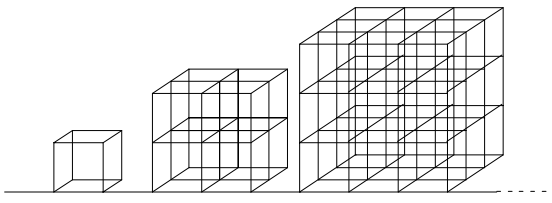


Figure 3: A graph with a square growth between C_1 and C_3 and not comparable with C_2

Many graphs are in none of the C_n like F_2 the free group with 2 generators (which has an exponential growth), and also graphs which can be plunged into $\mathbb{R}^{n+1}$ but not into $\mathbb{R}^n$ and which does not cover $\mathbb{R}^{n+1}$. For instance the graph of the figure 3 can be plunged into $\mathbb{R}^3$, but not into $\mathbb{R}^2$ and does not cover $\mathbb{R}^3$. Each graph of automata defined on this graph can be simulated by a GA defined on $\mathbb{Z}^3$. In addition, this graph simulates C_1 and is not comparable with C_2 although they have the same graph growth. The same idea allows us to build a graph simulated by $\mathbb{Z}^m$, incomparable with $C_i, n < i < m$ and simulating $\mathbb{Z}^n$: we link growing balls of $\mathbb{Z}^m$ like on the previous figure and connect it to the graph $\mathbb{Z}^n$.

Szuzsanna Róka proved in [10, 11] that all the graphs $F_i, i > 1$ (i.e. the free groups with more than two generators) were equivalent for the unitary simulation. We denote their class by F. The following theorem proves that all the graphs can be simulated by a bounded degree (by d) tree and so by all the F_i ($i > 1$) because it is easy to inject this tree into F_d. Hence the F class is the maximum element for the simulation order. As a corollary, it proves that for all n, $C_n \leq F$.

Theorem 4.1 *Let G be a connected bounded degree (by d) graph. A developed graph of G is a tree with bounded degree (by d) and it simulates G (the simulation is not necessarily unitary).*

A developed graph is defined as follows:

Definition 4.1 *Let O be a generic vertex of G, the developed graph from O, $\mathcal{D}_O(G) = (V', E')$, of G is defined as follows: V' is the set of all the finite paths starting at O without going back (i.e. which have no subpath of the form: xyx). There is an edge between a and b if and only if they are paths like $a = (s_1, s_2, ..., s_n)$ and $b = (s_1, s_2, ..., s_{n+1})$, i.e. one is an initial segment of the other with exactly one more vertex.*

Let us remark that the developed graph of $\mathbb{Z}^2$ is F_2, and that more generally developed graphs of Cayley graphs are the free groups. So the simulation of $\mathbb{Z}^2$ by F_2 works exactly like in the following proof. Each vertex of F_2 has the same neighborhood than the simulated vertex of $\mathbb{Z}^2$ so each vertex is simulated by an infinite number of vertex of F_2.

Proof 4.1 Each developed graph is clearly a tree whose degree in $x = (s_1, ..., s_n)$ is equal to s_n degree in G because x is linked to the vertex $(s_1, ..., s_{n-1})$ and to each vertex $(s_1, ..., s_n, s)$ where s is a neighbor of s_n and $s \neq s_{n-1}$ because going back is forbidden.

Let π be the canonical projection of $\mathcal{D}_O(G)$ into G (which associates to each path its last vertex). We will simulate the vertex v by all the vertices $\pi^{-1}(v)$ in $\mathcal{D}_O(G)$. This simulation is not necessarily unitary if for some v $\pi^{-1}(v)$ is infinite. The simulating automaton has exactly the same states and the same transition function. The neighborhood of each vertex in G is preserved by π so the evolution of the states in v and in each vertex of $\pi^{-1}(v)$ is the same. The simulation is done without lost of time. □

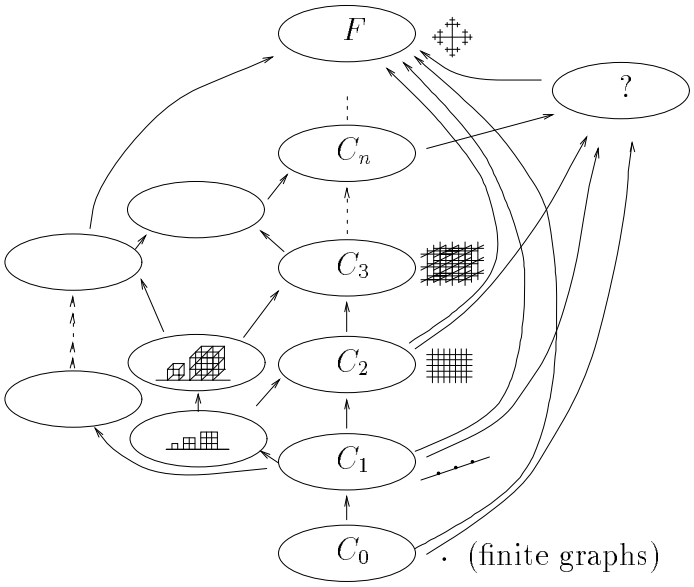


Figure 4: The graph hierarchy with relation to graphs of automata simulation

The hierarchy of graphs w.r.t. the simulation order can be represented as on the figure 4. Except C_0, the class of finite graphs (which contains the finite automata), C_1 is smallest that all the other classes because they all contain a connected infinite graph and so an infinite path (and $\mathbb{N} \in C_1$). The class F is the maximum element of our order. All the other classes are greater than C_1, smaller than F and may be incomparable with some C_n. There exist classes simulating C_j, simulated by C_k and incomparable with the classes C_i, $j < i < k$ and classes simulating C_k and incomparable with the classes C_i, $i > k$. At last, between two classes C_i and C_j (except C_0 and C_1), there are an infinite number of classes.

4.2 Study of the Maximum Class

We will now study more precisely the class F. All the graphs in F have an exponential growth (else they could not simulate F_2) and so none can be plunged in any $\mathbb{R}^n$. Hence we will consider an other interesting metric space with compact closed balls: the 2-dimensional hyperbolic space H_2. Here is one of the definitions of H_2:

Definition 4.2 *H_2 is the unitary radius opened disk with the Riemann-metric defined by its scalar product at the point x by:*

$$g_x(v,w) = \frac{< v, w >}{(1 - ||x||^2)^2}$$

In this space, the perimeter of a circle with radius r is:

$$P = \frac{\pi(e^{2r} - 1)(e^{2r} + 1)}{2e^{2r}} \sim \frac{\pi}{2} e^{2r}$$

and so grows exponentially with the radius. So this space seems interesting to study F. A first application is to prove that F_2 can be plunged into H_2:

Theorem 4.2 *F_2 can be plunged into H_2.*

Proof 4.2 Let O be an arbitrary point of H_2. Let us consider all the points of F_2 whose distance to origin is n, we will put them regularly on the circle centered in O with perimeter $4 \times 3^{n-1}$ like on the figure 5. Let r_n be the radius of the circle of perimeter $4 \times 3^{n-1}$, $r_{n+1} - r_n$ is bounded (i.e. $\forall n \;\; m \leq r_{n+1} - r_n \leq M$). So the distance between two neighbors is less than $M + 1$ and more than m, this is a plunging into H_2. □

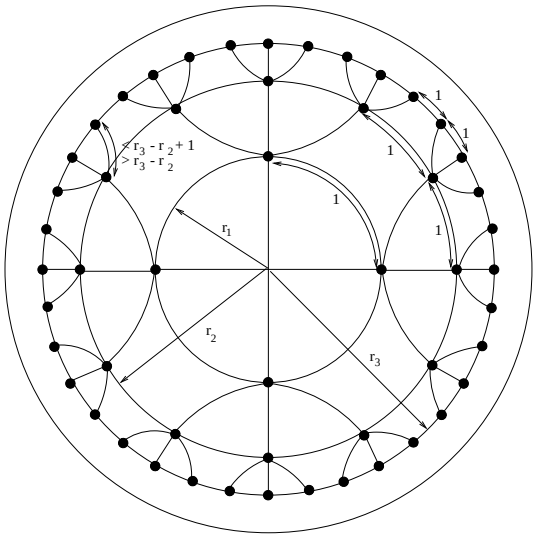


Figure 5: The plunging of F_2 into H_2

So all the tilings of H_2 are of course equivalent (for the unitary simulation) and are also in F. Let us now give some examples of tiling of H_2.

A regular polygon with k vertices is called a k-gon. Let $\Gamma(k,d)$ $(k,d \geq 3)$ be the graph such as each vertex is surrounded by d k-gons (see [1]). These graphs are studied with relation to this simulation in [8]. For instance $\Gamma(4,3)$ is the cube (which is in C_0), $\Gamma(6,3)$ is the hexagonal tiling of the plan (and is in C_2). If $k = 3$ and $d \geq 7$, $k = 4$ and $d \geq 5$, $k = 5$ and $d \geq 4$, $k = 6$ and $d \geq 4$ or $k \geq 7$ then the graph cannot be embedded in any $\mathbb{R}^n$ because the angle sum in each vertex is greater than 2π.

In fact all these graphs are tilings of H_2. H_2 is the typical example of space with constant curvature -1. So the sum of a triangle angles is given by: $\hat{A} + \hat{B} + \hat{C} = \pi(1 - \mathcal{A}/\pi)$, where $\mathcal{A}$ is the triangle area, the sum is also between 0 and its value in Euclidean spaces: π. So for a k-gon, we have $\hat{A_1} + ... + \hat{A_k} = (k-2)\pi(1 - A/(k-2)\pi)$. Hence, we can chose the scale of the k-gon such that its angles are $2\pi/d$, so we obtain a tiling.

Our theorem proves that all these $\Gamma(k,d)$ which cannot be embedded in any $\mathbb{R}^n$ are in F and equivalent for the unitary simulation.

5 Discussion and Open Problems

The hierarchy of all the connected bounded degree graphs according to the intuitive notion of simulation defined in [10, 11] is not completely known, but we can classify most of the graphs because when we imagine a graph we naturally embed it in a known space.

In particular, we do not know whether a class different from F simulates all the classes C_n.

In addition, the link between graph growth and the hierarchy is enlightened because tilings of $\mathbb{R}^n$ spaces have an asymptotic growth in x^n and all the graphs in F have an exponential growth.

A graph cannot simulate another graph with a higher growth, but having a higher growth is not enough to simulate a graph (see figure 3). A natural question is to ask whether a "regular" graph can simulate all the graphs of smaller growth, and a first step would be to consider Cayley graphs (for a general introduction to Cayley graphs, see for instance [6]) and many questions are open:

Among finitely generated Cayley graphs (but with an infinite number of relations), some have neither polynomial, nor exponential growth (Grigorchuk [3], reformed and applied to language recognition by [2]), where are they in the hierarchy?

For finite presentation Cayley graphs, the existence of such graphs is still an open problem. So maybe all finitely presented Cayley graphs are in F or a C_n. This raises the two following questions:

- The underlying groups of Cayley graphs with polynomial growth are exactly virtually nilpotent groups [4]. These groups include commutative ones ($\mathbb{Z}^n$) and graphs like $U_3(\mathbb{Z})$ (the unipotent matrices like
$$\begin{pmatrix} 1 & a & b \\ 0 & 1 & c \\ 0 & 0 & 1 \end{pmatrix}$$
where a, b, $c \in \mathbb{Z}$, see figure 6), which has 3 generators but its graph growth is equivalent to x^4. Is this graph equivalent to $\mathbb{Z}^4$?

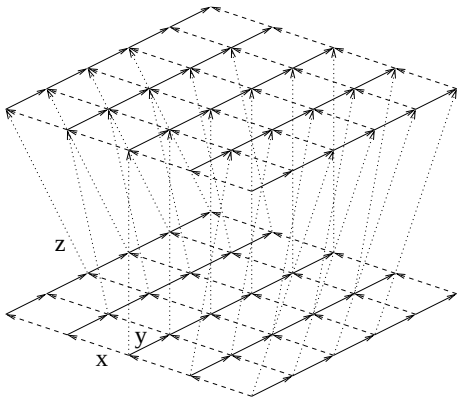


Figure 6: $U_3(\mathbb{Z})$ group: $< x, y, z \mid xy = yx, yz = zy, zx = xzy >$

- Are all exponential growth Cayley graphs in F?

A last interesting problem would be to find a more restrictive simulation to classify more precisely the graphs of F.

Conclusion

The main structure of the hierarchy of the graphs with relation to the power of the graphs of automata that can be defined on them is now well known. Anyway, many problems remain open, the power of Cayley graphs like $U_3(\mathbb{Z})$ or Grigorchuk examples are still unknown.

Acknowledgment

The author thanks Sylvie Ruette for many useful remarks, careful proof checking and her support. He also wishes to thank Bill Allombert for numerous and fruitful discussions and one of the anonymous referee for many helpful comments.

References

[1] T. Chaboud and C. Kenyon. Planar cayley graphs with regular duals. *International Journal of Algebra and Computation*, 6(5):553–561, 1996.

[2] Max Garzon and Yechezkel Zalcstein. The complexity of Grigorchuk groups with application to cryptography. *Theoretical Computer Science*, 88(1):83–98, 30 September 1991.

[3] R. Grigorchuk. Degrees of growth of finitely generated groups and the theory of invariant means. *Math. USSR Izvestiya*, 25:259–300, 1985.

[4] M. Gromov. Groups of polynomial growth and expanding maps. *Publ. Math. I.H.E.S.*, 53:53–78, 1981.

[5] Pascal Koiran, Michel Cosnard, and Max Garzon. Computability with low-dimensional dynamical systems. In *STACT'93*, volume 665 of *Lecture Notes in Computer Science*, pages 365–373, Würzburg, Germany, 25–27 February 1993. Springer-Verlag.

[6] R. C. Lyndon and P. E. Schupp. *Combinatorial Group Theory*. Springer Verlag, 1977.

[7] J. Mazoyer and I. Rapaport. Inducing an order on cellular automata by a grouping operation. In *STACS'98*, volume 1373, pages 116–127. Lecture Notes in Computer Science, 1998.

[8] C. Nichitiú. Simulation of graph automata. *preprint*, 1998.

[9] E. Rémila. Recognition of graphs by automata. *Theoretical Computer Science*, 136:291–332, 1994.

[10] Zs. Róka. *Automates cellulaires sur graphes de Cayley*. PhD thesis, Ecole Normale Supérieure de Lyon, 1994.

[11] Zs. Róka. Simulations between cellular automata on cayley graphs. *Theoretical Computer Science*, to appear.

[12] P. Rosensthiel, J. R. Fiksel, and A. Holliger. Intelligent graphs: Networks of finite automata capable of solving graph problems. In R. C. Reed, editor, *Graph Theory and computing*, pages 210–265. Academic Press, 1973.

Number-Conserving Reversible Cellular Automata and Their Computation-Universality*

Kenichi MORITA, and Katsunobu IMAI

Hiroshima University, Faculty of Engineering
Higashi-Hiroshima, 739-8527, Japan
{morita, imai}@ke.sys.hiroshima-u.ac.jp

Abstract

We introduce a new model of cellular automaton called a one-dimensional number-conserving partitioned cellular automaton (NC-PCA). An NC-PCA is a system such that a state of a cell is represented by a triple of non-negative integers, and the total (i.e., sum) of integers over the configuration is conserved throughout its evolving (computing) process. It can be thought as a kind of modelization of the physical conservation law of mass (particles) or energy. We also define a reversible version of NC-PCA, and prove that a reversible NC-PCA is computation-universal. It is proved by showing that a reversible two-counter machine, which has been known to be universal, can be simulated by a reversible NC-PCA.

1 Introduction

Recently, various kinds of interesting computing models which directly reflect laws of nature have been proposed and investigated. Among others, quantum computing, DNA computing, reversible computing, etc. have been extensively studied. A reversible computer is a system such that its transition function of the whole state is a one-to-one mapping (injection), hence, roughly speaking, it is a backward deterministic system. It is a kind of model reflecting physical reversibility, and has been known to be very important when studying inevitable power dissipation in a computing process [3, 4]. In spite of the constraint of reversibility, such a system has rich ability of computing. Bennett first showed computation-universality of a reversible Turing machine [2]. A reversible cellular automaton (CA) has also been studied extensively, and several versions of universality results have been shown [7, 8, 9, 10, 11, 13, 14].

Conservation of mass or energy is also an important physical law as well as reversibility. Fredkin and Toffoli [4] proposed Conservative Logic, a kind of logic circuit theory, that models both reversibility and conservation law of physics, and showed its universality. In this system, each primitive logic gate must satisfy the constraints of reversibility (i.e., its logical function is an injection), and conservation of bits (i.e., the total number of logical value "1"s is conserved between its input and output). Also for cellular automata, several universal models that are both reversible and bit-conserving have been known [7, 8, 10].

In this paper, we define a new model of cellular automaton (CA) called a one-dimensional number-conserving partitioned cellular automaton (NC-PCA), which generalize the notion of bit-conserving CA. In an NC-PCA, each cell is partitioned into three parts, i.e., left, center, and right parts, and the state of each part is represented by a non-negative integer (thus, the state of a cell is represented by a triple of non-negative integers). The next state of a cell is determined

*This work was supported in part by Grant-in-Aid for Scientific Research (C), No. 10680355, from Ministry of Education, Science, Sports and Culture of Japan.

by the present states of the right part of the left-neighboring cell, the center part of this cell, and the left part of the right-neighboring cell (not depending on the whole state of the three cells). The total number is conserved during the local transition, hence the total number over a configuration is also conserved throughout the evolving process.

Related to this model, a few other models in which each cell state is represented by a non-negative integer have been known: a totalistic CA [1] and a sand pile model [5, 6]. In [1], a CA with a simple totalistic rule (but not necessarily number-conserving) has been shown to be universal. In [5, 6], a kind of an automata system having a specific type of number-conserving rules are studied.

Here, we investigate the computing ability of an NC-PCA, and its reversible version. We show that an NP-PCA is computation universal even if it is reversible. This strengthens the previous result that a one-dimensional reversible CA (not necessarily a number-conserving one) is computation-universal [9]. We prove it by showing that a reversible two-counter machine, which has been known to be universal [12], can be simulated by a reversible NC-PCA.

2 Number-Conserving Partitioned Cellular Automata

In order to define a one-dimensional number-conserving partitioned cellular automaton (NC-PCA), we first give a definition of a partitioned cellular automaton (PCA) that has been introduced to design a reversible cellular automaton [9].

Definition 2.1 A *deterministic one-dimensional three-neighbor partitioned cellular automaton* (PCA) is a system defined by

$$A = (\mathbf{Z}, (L, C, R), g, (\tilde{q}_L, \tilde{q}_C, \tilde{q}_R)),$$

where $\mathbf{Z}$ is the set of all integers at which cells are placed, L, C and R are non-empty finite sets of states of left, center, and right parts of a cell, $g : R \times C \times L \to L \times C \times R$ is a local function, and $(\tilde{q}_L, \tilde{q}_C, \tilde{q}_R) \in L \times C \times R$ is a quiescent state that satisfies $g(\tilde{q}_R, \tilde{q}_C, \tilde{q}_L) = (\tilde{q}_L, \tilde{q}_C, \tilde{q}_R)$.

A *configuration* over the set $Q = L \times C \times R$ is a mapping $\alpha : \mathbf{Z} \to Q$. Let $\mathrm{Conf}(Q)$ denote the set of all configurations over Q, i.e., $\mathrm{Conf}(Q) = \{\alpha \mid \alpha : \mathbf{Z} \to Q\}$. A *quiescent configuration* is the one such that all the cells are in the quiescent states $(\tilde{q}_L, \tilde{q}_C, \tilde{q}_R)$.

Let $\mathrm{pro}_L : Q \to L$ is a projection function such that $\mathrm{pro}_L(l, c, r) = l$ for all $(l, c, r) \in Q$. Projection functions $\mathrm{pro}_C : Q \to C$, and $\mathrm{pro}_R : Q \to R$ are also defined similarly. The *global function* $G : \mathrm{Conf}(Q) \to \mathrm{Conf}(Q)$ of A is defined as follows.

$$\forall x \in \mathbf{Z} : \; G(\alpha)(x) = g(\mathrm{pro}_R(\alpha(x-1)), \mathrm{pro}_C(\alpha(x)), \mathrm{pro}_L(\alpha(x+1)))$$

Fig. 1 shows how the local function g is applied to each cell. In the following, an equation $g(r, c, l) = (l', c', r')$ is called a *rule* of A, and write it by

$$[r, c, l] \to [l', c', r'].$$

We regard the local function g as the set of such rules for convenience.

Next, we define the notion of reversibility for PCAs.

Definition 2.2 Let $A = (\mathbf{Z}, (L, C, R), g, (\tilde{q}_L, \tilde{q}_C, \tilde{q}_R))$ be a PCA. We say A is *globally reversible* iff its global function G is one-to-one, and *locally reversible* iff its local function g is one-to-one.

It is easy to prove the following proposition on PCA, which has been shown in [9].

Proposition 2.1 Let A be a PCA. A is globally reversible iff it is locally reversible.

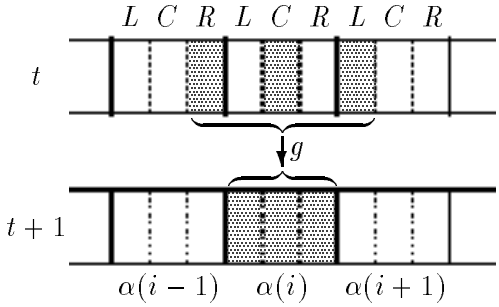


Figure 1: The local transition function g of a PCA A.

By Proposition 2.1, a globally or locally reversible PCA is called simply "reversible" and denoted by RPCA. By this, if we want to construct a reversible CA, it is sufficient to give a PCA whose local function g is one-to-one. This makes it easy to design a reversible CA.

When we design a one-to-one local function g, it is sufficient to define it only on a subset of $R \times C \times L$ that are needed to perform a given task. Because, we can always find a one-to-one extension from a given partial function provided that the latter function is one-to-one on the subset. This is assured by the following proposition (its proof is omitted since it is easy).

Proposition 2.2 Let A and B be finite sets such that $|A| = |B|$, and let g be a mapping $A' \to B$ for some A' ($\subset A$). If g is one-to-one, then there is a one-to-one mapping $g' : A \to B$ such that $g'(a) = g(a)$ for all $a \in A'$.

We now give a definition of a number-conserving PCA. As in the case of a reversible CA, it is also convenient to use the framework of a PCA. Because, the the notion of number-conservation can be expressed by a simple constraint on a local function of a PCA.

Definition 2.3 Let $A = (\mathbf{Z}, (\mathbf{N}_m, \mathbf{N}_m, \mathbf{N}_m), g, (0, k, 0))$ be a PCA, where $\mathbf{N}_m$ denotes the set of integers $\{0, 1, \cdots, m-1, m\}$, and $k(\leq m)$ is a non-negative integer. A is called a *one-dimensional number-conserving partitioned cellular automaton* (NC-PCA), iff it satisfies the following condition: For all (r, c, l), $(l', c', r') \in \mathbf{N}_m^3$, if $g(r, c, l) = (l', c', r')$, then $r + c + l = l' + c' + r'$.

A *reversible NC-PCA* is also defined similarly, and denoted by NC-RPCA.

Example 2.1 A simple example of an NC-RPCA:

$$A_1 = (\mathbf{Z}, \mathbf{N}_2^3, g_1, (0, 0, 0)).$$

The local function g_1 contains the following rules.

[0,0,0] → [0,0,0]	[1,1,0] → [0,0,2]	[2,1,0] → [2,1,0]
[0,1,0] → [0,1,0]	[0,1,1] → [2,0,0]	[0,1,2] → [0,1,2]
[1,0,0] → [0,0,1]	[2,0,0] → [1,1,0]	
[0,0,1] → [1,0,0]	[0,0,2] → [0,1,1]	

We can verify that each rule satisfies the constraint of number-conservation. It is also easy to see that the right-hand side of each rule differs from those of the others, hence A_1 is reversible. Fig. 2 shows an example of its transitions of configurations, where each number is represented by this number of particles. We can observe that single "flying particle" goes back and forth between the "walls" made also of particles. Each time the flying particle collides a wall, the latter is shifted by one cell.

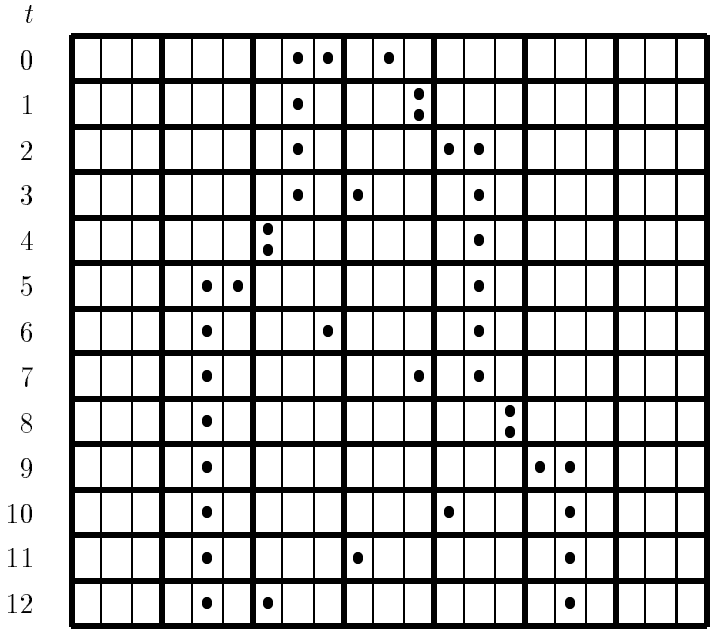


Figure 2: Behavior of the NC-RPCA A_1.

3 Universality of an NC-RPCA

In this section, we show that for any reversible two-counter machine there is an NC-RPCA that simulates it. Since a reversible two-counter machines has been known to be computation-universal [12], we can conclude that an NC-RPCA is also universal.

In [12] a counter machine (CM) is defined as a kind of multi-tape Turing machine whose heads are read-only ones and whose tapes are all blank except the leftmost squares as shown in Fig. 3 (P is a blank symbol). This definition is convenient for giving the notion of reversibility on a CM.

Definition 3.1 A *k-counter machine* (CM(k)) is a system

$$M = (k, Q, \delta, q_0, q_f),$$

where k is the number of tapes (or counters), Q is a nonempty finite set of internal states, $q_0 \in Q$ is an initial state, and $q_f \in Q$ is a final (halting) state. M uses $\{Z, P\}$ as a tape alphabet. δ is a move relation which is a subset of $(Q \times \{0, 1, \cdots, k-1\} \times \{Z, P\} \times Q) \cup (Q \times \{0, 1, \cdots, k-1\} \times \{-, 0, +\} \times Q)$ (where "$-$", "0", and "$+$" denote left-shift, no-shift, and right-shift of a head, respectively). Tapes are one-way (rightward) infinite. The leftmost squares of the tapes contain the symbol "Z"s, and all the other squares contain "P"s (Z and P stand for "zero" and "positive").

Each element of δ is called a *quadruple*, and is either of the form

$$[q, i, s, q'] \quad \text{or} \quad [q, i, d, q'],$$

where $q, q' \in Q$, $i \in \{0, 1, \cdots, k-1\}$, $s \in \{Z, P\}$, $d \in \{-, 0, +\}$. The quadruple $[q, i, s, q']$ means that if M is in the state q and the i-th head is reading the symbol s then change the state into q'. It is used to test whether the contents of a counter are zero or positive. On the other hand, $[q, i, d, q']$ means that if M is in the state q then shift the i-th head to the direction d and change

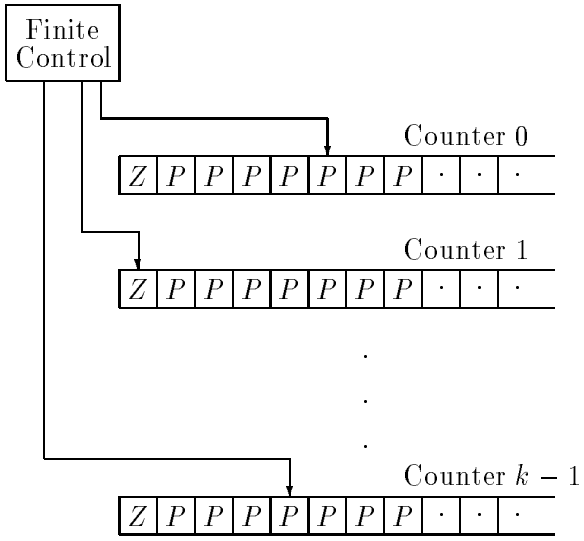


Figure 3: A k-counter machine (CM(k)).

the state into q'. It is used to increment or decrement a counter by one (or make no change if $d = 0$).

Definition 3.2 An *instantaneous description* (ID) of a CM(k) $M = (k, Q, \delta, q_0, q_f)$ is a $(k+1)$-tuple

$$(q, n_0, n_1, \cdots, n_{k-1}) \in Q \times \mathbf{N}^k,$$

where $\mathbf{N} = \{0, 1, \cdots\}$. It represents that M is in the state q and the counter i keeps n_i (we assume the position of the leftmost square of a tape is 0). The transition relation $\vdash_M$ over IDs of M is defined as follows:

$$\begin{array}{l} (q, n_0, \cdots, n_{i-1}, n_i, n_{i+1}, \cdots, n_{k-1}) \\ \vdash_M (q', n_0, \cdots, n_{i-1}, n_i', n_{i+1}, \cdots, n_{k-1}) \end{array}$$

holds iff one of the following conditions (1)–(5) is satisfied.

(1) $[q, i, Z, q'] \in \delta$ and $n_i = n_i' = 0$.
(2) $[q, i, P, q'] \in \delta$ and $n_i = n_i' > 0$.
(3) $[q, i, -, q'] \in \delta$ and $n_i - 1 = n_i'$.
(4) $[q, i, 0, q'] \in \delta$ and $n_i = n_i'$.
(5) $[q, i, +, q'] \in \delta$ and $n_i + 1 = n_i'$.

We denote reflexive and transitive closure of $\vdash_M$ by $\vdash_M^*$, and n-step transition by $\vdash_M^n$ ($n = 0, 1, \cdots$).

Definition 3.3 Let $M = (k, Q, \delta, q_0, q_f)$ be a CM(k), and

$$\alpha_1 = [p_1, i_1, x_1, p_1'] \quad \text{and} \quad \alpha_2 = [p_2, i_2, x_2, p_2']$$

be two distinct quadruples in δ. We say α_1 and α_2 *overlap in domain* iff the following holds, where $D = \{-, 0, +\}$.

$$p_1 = p_2 \ \wedge \ [i_1 \neq i_2 \ \vee \ x_1 = x_2 \ \vee \ x_1 \in D \ \vee \ x_2 \in D]$$

We say α_1 and α_2 *overlap in range* iff the following holds.

$$p_1' = p_2' \ \wedge \ [i_1 \neq i_2 \ \vee \ x_1 = x_2 \ \vee \ x_1 \in D \ \vee \ x_2 \in D]$$

A quadruple α is called *deterministic* (*reversible*, respectively) iff there is no other quadruple in δ which overlaps in domain (range) with α. M is called *deterministic* (*reversible*, respectively) iff every quadruple in δ is deterministic (reversible). A reversible CM(k) is denoted by RCM(k).

For example, the following pair

$$[q_1, 2, P, q_3] \ \text{ and } \ [q_4, 2, +, q_3]$$

overlaps in range, while the pair

$$[q_1, 2, Z, q_3] \ \text{ and } \ [q_4, 2, P, q_3]$$

does not. As seen from this definition, every ID of a deterministic (reversible, respectively) CM(k) has at most one ID that immediately follows (precedes) it. Hereafter, we consider only deterministic reversible and deterministic irreversible CM(k)s.

It has been known that an RCM(2) is computation-universal [12].

Proposition 3.1 [12] For any Turing machine T, there is a deterministic RCM(2) M that simulates T.

We need the following Lemma to prove Theorem 3.1

Lemma 3.1 For any deterministic CM(2) $M = (2, Q, \delta, q_0, q_f)$, there is a deterministic CM(2) $M' = (2, Q', \delta, q_0', q_f')$ that simulates M satisfying the following conditions: (i) The initial state q_0' never appears as the fourth element of a quadruple in δ' (hence it appears only at time 0). (ii) If M is reversible then M' is also reversible.

Proof. In the case M is irreversible, it is very easy to construct such M' by adding a new initial state to Q. So, we consider the reversible case. In [12], a construction method of a reversible CM(2) M_2 that simulates a given CM(k) M_1 ($k = 1, 2, \cdots$) (that is not necessarily reversible) has been shown. By checking the construction method shown in [12], we can verify that M_2 satisfies the above condition (i), provided that M_1 also satisfies it. Hence the Lemma holds. □

Theorem 3.1 For any deterministic CM(2) M, there is a deterministic NC-PCA A that simulates M satisfying the following condition: If M is reversible then A is also reversible.

Proof. Without loss of generality, we assume that the state set of M is $Q = \{q_0, q_1, \cdots, q_{m-1}\}$, and the initial and final states are q_0, and q_{m-1}, respectively. Hence,

$$M = (2, \{q_0, q_1, \cdots, q_{m-1}\}, \delta, q_0, q_{m-1}).$$

Further assume that q_0 never appears as the fourth element of a quadruple in δ (by Lemma 3.1). Let Inc_j, Dec_j, Nop, and Test_j be the sets of states defined as follows ($j \in \{0, 1\}$).

$$\begin{array}{lcl}
\text{Inc}_j & = & \{q_i \mid [q_i, j, +, q_k] \in \delta \text{ for some } q_k \in Q\} \\
\text{Dec}_j & = & \{q_i \mid [q_i, j, -, q_k] \in \delta \text{ for some } q_k \in Q\} \\
\text{Nop} & = & \{q_i \mid [q_i, j, 0, q_k] \in \delta \text{ for some } j \in \{0, 1\}, \text{ and } q_k \in Q\} \\
\text{Test}_j & = & \{q_i \mid [q_i, j, s, q_k] \in \delta \text{ for some } s \in \{Z, P\}, \text{ and } q_k \in Q\}
\end{array}$$

These sets stand for instructions of "increment the counter j", "decrement the counter j", "no-operation", and "test if the counter j is zero or positive". It is easy to see that Inc_j, Dec_j, Nop, and Test_j are pairwise disjoint (for example, $\mathrm{Inc}_0 \cap \mathrm{Inc}_1 = \emptyset$, $\mathrm{Dec}_1 \cap \mathrm{Nop} = \emptyset$, etc.), since M is deterministic.

We now construct an NC-PCA A that simulates M. Each part of a cell of A keeps a number at most $m + 18$, and the quiescent state is $(0, 0, 0)$. Thus,

$$A = (\mathbf{Z}, \mathbf{N}^3_{m+18}, g, (0, 0, 0)).$$

Before defining the local function g, we fix a coding method of an ID of M by a configuration of A. First, we assign an "operation code" to each state of M by the following function $\gamma : Q \to \{0, 2, 4, 6, 7\}$.

$$\gamma(q_i) = \begin{cases} 2 & \text{if } q_i \in \mathrm{Inc}_0 \\ 4 & \text{if } q_i \in \mathrm{Dec}_0 \\ 6 & \text{if } q_i \in \mathrm{Inc}_1 \\ 7 & \text{if } q_i \in \mathrm{Dec}_1 \\ 0 & \text{otherwise} \end{cases}$$

Note that the states in $\mathrm{Nop} \cup \mathrm{Test}_0 \cup \mathrm{Test}_1$ and the halting states have the same operation code 0. We then define a coding function $\varphi : Q \times \mathbf{N}^2 \to \mathrm{Conf}(\mathbf{N}^3_{m+18})$, which maps each ID of M to a configuration of A. Let $I = (q_i, n_0, n_1)$ be an ID of M. A configuration $\varphi(I)$ is computed by the following procedure.

begin
  $\alpha :=$ the quiescent configuration;
  $\mathrm{pro}_L(\alpha(0)) := i + 10$;
  $\mathrm{pro}_C(\alpha(0)) := (m + 16) - (i + 10) - \gamma(q_i)$;
  $\mathrm{pro}_R(\alpha(0)) := \gamma(q_i)$;
  for each $j \in \{0, 1\}$ **do**
    if $n_j = 0$ and $q_i \in \mathrm{Inc}_j$ **then** $\mathrm{pro}_R(\alpha(0)) := \mathrm{pro}_R(\alpha(0)) + 2^j$
    else $\mathrm{pro}_C(\alpha(n_j)) := \mathrm{pro}_C(\alpha(n_j)) + 2^j$;
  $\varphi(I) := \alpha$
end.

For example, the configuration $\varphi(q_i, 2, 0)$ such that $q_i \in \mathrm{Inc}_1$ is shown in Fig. 4.

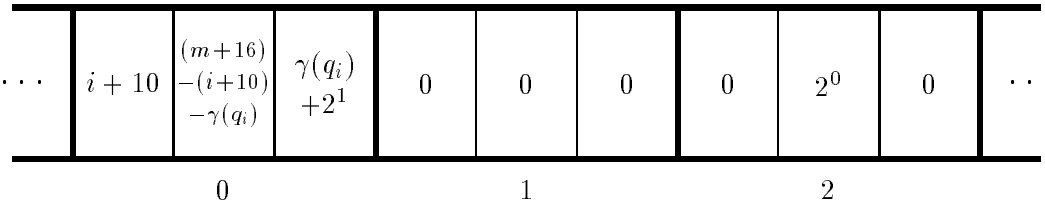


Figure 4: The configuration $\varphi(q_i, 2, 0)$ such that $q_i \in \mathrm{Inc}_1$.

Each configuration $\varphi(I)$ has the number $m + 19$ in total. The number n_j kept by the counter j ($j \in \{0, 1\}$) is recorded by putting the number 2^j at the center part of the cell n_j (at the right part of the cell 0, if $n_j = 0$ and the current state is in Inc_j). The number 2^j is called a *counter marker*. The remaining $m + 16$ particles are used to record the state of M, and to execute operations on counters.

In what follows, each state in $\{1, 2, \cdots, m+9\}$ appearing in the left or the right part (not in the center part) of a cell is called a *signal*. Signals in $\{10, 11, \cdots, m+9\}$ are called *state signals*, and are used to record the current state of M. State signals are kept by the cells 0 and -1 (they go back and forth between these cells). Signals in $\{2, 4, 6, 7\}$ are called *operation signals*, which are used to execute increment/decrement operations. Each of the four operation signals sometimes carry a counter marker to move it to the right- or left-neighboring cell. At that time, these signals "2", "4", "6", and "7" temporarily become "3", "5", "8", and "9", respectively. The signal "1" is a special one called an *initial/final signal* (it will be explained later).

We now define the local function g of A as follows.

1. Rules for the cases where no signal exists:
For each $x \in \mathbf{N}_{m+18}$, include the following rule in g.

$$[\ 0,\ x,\ 0\ ] \to [\ 0,\ x,\ 0\ ] \tag{1}$$

2. Rules for state signals:
For each $x \in \{10, 11, \cdots, m+9\}$, include the following rule in g.

$$[\ 0,\ 0,\ x\ ] \to [\ 0,\ 0,\ x\ ] \tag{2}$$

3. Rules for the increment operation:
For each $j \in \{0, 1\}$ and $c \in \{0, 1\}$, include the following rules in g ($\oplus$ denotes the addition in mod 2).

$$\begin{array}{lll}
[\ 2+4j, & c\cdot 2^{j\oplus 1}, & 0\ ] \to [\ 0, & c\cdot 2^{j\oplus 1}, & 2+4j\ ] \quad (3.1)\\
[\ 2+4j, & 2^j + c\cdot 2^{j\oplus 1}, & 0\ ] \to [\ 0, & c\cdot 2^{j\oplus 1}, & 2+4j+2^j\ ] \quad (3.2)\\
[\ 2+4j+2^j, & c\cdot 2^{j\oplus 1}, & 0\ ] \to [\ 2+4j, & 2^j + c\cdot 2^{j\oplus 1}, & 0\ ] \quad (3.3)\\
[\ 0, & c\cdot 2^{j\oplus 1}, & 2+4j\ ] \to [\ 2+4j, & c\cdot 2^{j\oplus 1}, & 0\ ] \quad (3.4)
\end{array}$$

4. Rules for the decrement operation:
For each $j \in \{0, 1\}$ and $c \in \{0, 1\}$, include the following rules in g.

$$\begin{array}{lll}
[\ 4+3j, & c\cdot 2^{j\oplus 1}, & 0\ ] \to [\ 0, & c\cdot 2^{j\oplus 1}, & 4+3j\ ] \quad (4.1)\\
[\ 4+3j, & 2^j + c\cdot 2^{j\oplus 1}, & 0\ ] \to [\ 4+3j+2^j, & c\cdot 2^{j\oplus 1}, & 0\ ] \quad (4.2)\\
[\ 0, & c\cdot 2^{j\oplus 1}, & 4+3j+2^j\ ] \to [\ 4+3j, & 2^j + c\cdot 2^{j\oplus 1}, & 0\ ] \quad (4.3)\\
[\ 0, & c\cdot 2^{j\oplus 1}, & 4+3j\ ] \to [\ 4+3j, & c\cdot 2^{j\oplus 1}, & 0\ ] \quad (4.4)
\end{array}$$

5. Rules for waiting for the completion of the increment/decrement operation:
For each $[q_i, j, d, q_k] \in \delta$ such that $d \in \{+, -\}$, and for each $c \in \{0, 1\}$, include the following rule in g.

$$\begin{array}{l}
[\ i+10,\ (m+16)-(i+10)-\gamma(q_i)+c\cdot 2^{j\oplus 1},\ 0\ ] \to\\
\qquad [\ i+10,\ (m+16)-(i+10)-\gamma(q_i)+c\cdot 2^{j\oplus 1},\ 0\ ] \qquad (5)
\end{array}$$

6. Rules for simulating M's state transition from a state in Inc_j:
For each $[q_i, j, +, q_k] \in \delta$ and $c \in \{0, 1\}$, if $q_k \notin \mathrm{Inc}_{j\oplus 1}$, then include the following rule in g.

$$\begin{array}{l}
[\ i+10,\ (m+16)-(i+10)-\gamma(q_i)+c\cdot 2^{j\oplus 1},\ \gamma(q_i)\ ] \to\\
\qquad [\ k+10,\ (m+16)-(k+10)-\gamma(q_k)+c\cdot 2^{j\oplus 1},\ \gamma(q_k)\ ] \qquad (6.1)
\end{array}$$

For each $[q_i, j, +, q_k] \in \delta$ and $c \in \{0, 1\}$, if $q_k \in \mathrm{Inc}_{j\oplus 1}$, then include the following rule in g.

$$\begin{array}{l}
[\ i+10,\ (m+16)-(i+10)-\gamma(q_i)+c\cdot 2^{j\oplus 1},\ \gamma(q_i)\ ] \to\\
\qquad [\ k+10,\ (m+16)-(k+10)-\gamma(q_k),\ \gamma(q_k)+c\cdot 2^{j\oplus 1}\ ] \qquad (6.2)
\end{array}$$

7. Rules for simulating M's state transition from a state in Dec_j:
For each $[q_i, j, -, q_k] \in \delta$ and $c, c' \in \{0,1\}$, if $q_k \notin (\mathrm{Inc}_j \cup \mathrm{Inc}_{j\oplus 1})$, then include the following rule in g.

$$\begin{array}{l}[\ i+10,\ \ (m+16)-(i+10)-\gamma(q_i)+c'\cdot 2^{j\oplus 1},\ \ \gamma(q_i)+c\cdot 2^j\ ] \ \rightarrow \\ \qquad [\ k+10,\ \ (m+16)-(k+10)-\gamma(q_k)+c\cdot 2^j+c'\cdot 2^{j\oplus 1},\ \ \gamma(q_k)\ ]\end{array} \quad (7.1)$$

For each $[q_i, j, -, q_k] \in \delta$ and $c, c' \in \{0,1\}$, if $q_k \in \mathrm{Inc}_j$, then include the following rule in g.

$$\begin{array}{l}[\ i+10,\ \ (m+16)-(i+10)-\gamma(q_i)+c'\cdot 2^{j\oplus 1},\ \ \gamma(q_i)+c\cdot 2^j\ ] \ \rightarrow \\ \qquad [\ k+10,\ \ (m+16)-(k+10)-\gamma(q_k)+c'\cdot 2^{j\oplus 1},\ \ \gamma(q_k)+c\cdot 2^j\ ]\end{array} \quad (7.2)$$

For each $[q_i, j, -, q_k] \in \delta$ and $c, c' \in \{0,1\}$, if $q_k \in \mathrm{Inc}_{j\oplus 1}$, then include the following rule in g.

$$\begin{array}{l}[\ i+10,\ \ (m+16)-(i+10)-\gamma(q_i)+c'\cdot 2^{j\oplus 1},\ \ \gamma(q_i)+c\cdot 2^j\ ] \ \rightarrow \\ \qquad [\ k+10,\ \ (m+16)-(k+10)-\gamma(q_k)+c\cdot 2^j,\ \ \gamma(q_k)+c'\cdot 2^{j\oplus 1}\ ]\end{array} \quad (7.3)$$

8. Rules for simulating M's state transition from a state in Nop:
For each $[q_i, j, 0, q_k] \in \delta$ and $c, c' \in \{0,1\}$, if $q_k \notin (\mathrm{Inc}_0 \cup \mathrm{Inc}_1)$, then include the following rule in g.

$$\begin{array}{l}[\ i+10,\ \ (m+16)-(i+10)+c\cdot 2^0+c'\cdot 2^1,\ \ 0\ ] \ \rightarrow \\ \qquad [\ k+10,\ \ (m+16)-(k+10)-\gamma(q_k)+c\cdot 2^0+c'\cdot 2^1,\ \ \gamma(q_k)\ ]\end{array} \quad (8.1)$$

For each $[q_i, j, 0, q_k] \in \delta$ and $c, c' \in \{0,1\}$, if $q_k \in \mathrm{Inc}_j$ for some j, then include the following rule in g.

$$\begin{array}{l}[\ i+10,\ \ (m+16)-(i+10)+c\cdot 2^j+c'\cdot 2^{j\oplus 1},\ \ 0\ ] \ \rightarrow \\ \qquad [\ k+10,\ \ (m+16)-(k+10)+\gamma(q_k)+c'\cdot 2^{j\oplus 1},\ \ \gamma(q_k)+c\cdot 2^j\ ]\end{array} \quad (8.2)$$

9. Rules for simulating M's state transition from a state in Test_j:
For each $[q_i, j, Z, q_k] \in \delta$ and $c \in \{0,1\}$, if $q_k \notin (\mathrm{Inc}_j \cup \mathrm{Inc}_{j\oplus 1})$, then include the following rule in g.

$$\begin{array}{l}[\ i+10,\ \ (m+16)-(i+10)+2^j+c\cdot 2^{j\oplus 1},\ \ 0\ ] \ \rightarrow \\ \qquad [\ k+10,\ \ (m+16)-(k+10)-\gamma(q_k)+2^j+c\cdot 2^{j\oplus 1},\ \ \gamma(q_k)\ ]\end{array} \quad (9.1)$$

For each $[q_i, j, Z, q_k] \in \delta$ such that and $c \in \{0,1\}$, if $q_k \in \mathrm{Inc}_j$, then include the following rule in g.

$$\begin{array}{l}[\ i+10,\ \ (m+16)-(i+10)+2^j+c\cdot 2^{j\oplus 1},\ \ 0\ ] \ \rightarrow \\ \qquad [\ k+10,\ \ (m+16)-(k+10)-\gamma(q_k)+c\cdot 2^{j\oplus 1},\ \ \gamma(q_k)+2^j\ ]\end{array} \quad (9.2)$$

For each $[q_i, j, Z, q_k] \in \delta$ and $c \in \{0,1\}$, if $q_k \in \mathrm{Inc}_{j\oplus 1}$, then include the following rule in g.

$$\begin{array}{l}[\ i+10,\ \ (m+16)-(i+10)+2^j+c\cdot 2^{j\oplus 1},\ \ 0\ ] \ \rightarrow \\ \qquad [\ k+10,\ \ (m+16)-(k+10)-\gamma(q_k)+2^j,\ \ \gamma(q_k)+c\cdot 2^{j\oplus 1}\ ]\end{array} \quad (9.3)$$

For each $[q_i, j, P, q_k] \in \delta$ and $c \in \{0,1\}$, if $q_k \notin \mathrm{Inc}_{j\oplus 1}$, then include the following rule in g.

$$\begin{array}{l}[\ i+10,\ \ (m+16)-(i+10)+c\cdot 2^{j\oplus 1},\ \ 0\ ] \ \rightarrow \\ \qquad [\ k+10,\ \ (m+16)-(k+10)-\gamma(q_k)+c\cdot 2^{j\oplus 1},\ \ \gamma(q_k)\ ]\end{array} \quad (9.4)$$

For each $[q_i, j, P, q_k] \in \delta$ and $c \in \{0,1\}$, if $q_k \in \mathrm{Inc}_{j\oplus 1}$, then include the following rule in g.

$$\begin{array}{l}[\ i+10,\ \ (m+16)-(i+10)+c\cdot 2^{j\oplus 1},\ \ 0\ ] \ \rightarrow \\ \qquad [\ k+10,\ \ (m+16)-(k+10)-\gamma(q_k),\ \ \gamma(q_k)+c\cdot 2^{j\oplus 1}\ ]\end{array} \quad (9.5)$$

t	0	0	2	0	0	0	0	2	0	0	1	0	0	0	0
$t+1$	0	0	0	0	0	2	0	2	0	0	1	0	0	0	0
$t+2$	0	0	0	0	0	0	0	2	2	0	1	0	0	0	0
$t+3$	0	0	0	0	0	0	0	2	0	0	0	3	0	0	0
$t+4$	0	0	0	0	0	0	0	2	0	0	0	0	2	1	0
$t+5$	0	0	0	0	0	0	0	2	0	2	0	0	0	1	0
$t+6$	0	0	0	0	0	0	2	2	0	0	0	0	0	1	0
$t+7$	0	0	0	2	0	0	0	2	0	0	0	0	0	1	0

Figure 5: Performing an increment operation to the counter 0.

10. Rules for the initial/final signal:
Include the following rules in g.

$$[\ 1,\ 0,\ 0\ ] \rightarrow [\ 0,\ 0,\ 1\ ] \quad (10.1)$$
$$[\ 0,\ 0,\ 1\ ] \rightarrow [\ 1,\ 0,\ 0\ ] \quad (10.2)$$

For each $c, c' \in \{0,1\}$, if $q_0 \notin (\mathrm{Inc}_0 \cup \mathrm{Inc}_1)$, then include the following rule in g.

$$[\ 1,\ (m+15)+c\cdot 2^0+c'\cdot 2^1,\ 0\ ] \rightarrow [\ 10,\ (m+16)-10-\gamma(q_0)+c\cdot 2^0+c'\cdot 2^1,\ \gamma(q_0)\ ] \quad (10.3)$$

For each $c, c' \in \{0,1\}$, if $q_0 \in \mathrm{Inc}_j$ for some j, then include the following rule in g.

$$[\ 1,\ (m+15)+c\cdot 2^j+c'\cdot 2^{j\oplus 1},\ 0\ ] \rightarrow [\ 10,\ (m+16)-10-\gamma(q_0)+c'\cdot 2^{j\oplus 1},\ \gamma(q_0)+c\cdot 2^j\ ] \quad (10.4)$$

For each $c, c' \in \{0,1\}$, include the following rule in g.

$$[\ (m+9),\ (m+16)-(m+9)+c\cdot 2^0+c'\cdot 2^1,\ 0\ ] \rightarrow [\ 1,\ (m+15)+c\cdot 2^0+c'\cdot 2^1,\ 0\ ] \quad (10.5)$$

We now explain how each operation of M can be simulated by the rules of A. Although the simulation method itself is a rather straight-forward one, the above rules are designed so that the condition "if M is reversible, so is A" holds.

(a) Execution of an increment operation $[q_i, j, +, q_k]$:
The operation signals "2" and "6" are used for the increment of the counters 0 and 1, respectively. Such a signal is generated at the cell 0, and travels rightward until it meets a corresponding counter marker 2^0 or 2^1. The signal shifts the counter marker to the right by one cell, and then goes back to the cell 0. This operation can be performed by the rules (3.1)–(3.4) (strictly speaking, they are "rule schemes"). Fig. 5 shows an example of this process.

When the operation signal returns to the cell 0, the state transition from q_i to q_k in M is simulated by the rule (6.1) or (6.2) (depending on whether $q_k \in \mathrm{Inc}_{j\oplus 1}$ or not) in A. Fig. 6 (the case $n_j > 0$) and Fig. 7 (the case $n_j = 0$) show examples of the whole execution processes. The rules (1), (2), and (5) are also used during this operation.

t	0	0	0	$i+10$	$(m+16)$ $-(i+10)$ $-\gamma(q_i)$	$\gamma(q_i)$	0	2^j	0	0	0	0
$t+1$	0	0	$i+10$	0	$(m+16)$ $-(i+10)$ $-\gamma(q_i)$	0	0	0	$\gamma(q_i)$ $+2^j$	0	0	0
$t+2$	0	0	0	$i+10$	$(m+16)$ $-(i+10)$ $-\gamma(q_i)$	0	0	0	0	$\gamma(q_i)$	2^j	0
$t+3$	0	0	$i+10$	0	$(m+16)$ $-(i+10)$ $-\gamma(q_i)$	0	$\gamma(q_i)$	0	0	0	2^j	0
$t+4$	0	0	0	$k+10$	$(m+16)$ $-(k+10)$ $-\gamma(q_k)$	$\gamma(q_k)$	0	0	0	0	2^j	0

Figure 6: Execution of an increment operation $[q_i, j, +, q_k]$ for the case $n_j > 0$.

t	0	0	0	$i+10$	$(m+16)$ $-(i+10)$ $-\gamma(q_i)$	$\gamma(q_i)$ $+2^j$	0	0	0	0	0	0
$t+1$	0	0	$i+10$	0	$(m+16)$ $-(i+10)$ $-\gamma(q_i)$	0	$\gamma(q_i)$	2^j	0	0	0	0
$t+2$	0	0	0	$k+10$	$(m+16)$ $-(k+10)$ $-\gamma(q_k)$	$\gamma(q_k)$	0	2^j	0	0	0	0

Figure 7: Execution of an increment operation $[q_i, j, +, q_k]$ for the case $n_j = 0$.

t	0	0	7	0	0	0	0	0	0	0	3	0	0	0	0
$t+1$	0	0	0	0	0	7	0	0	0	0	3	0	0	0	0
$t+2$	0	0	0	0	0	0	0	0	7	0	3	0	0	0	0
$t+3$	0	0	0	0	0	0	0	0	0	9	1	0	0	0	0
$t+4$	0	0	0	0	0	0	7	2	0	0	1	0	0	0	0
$t+5$	0	0	0	7	0	0	0	2	0	0	1	0	0	0	0
$t+6$	7	0	0	0	0	0	0	2	0	0	1	0	0	0	0

Figure 8: Performing a decrement operation to the counter 1.

(b) Execution of a decrement operation $[q_i, j, -, q_k]$:
The operation signals "4" and "7" are used for the decrement of the counter 0 and 1, respectively. The shifting operation of a counter marker is similar to the case of the increment operation, and is be performed by the rules (4.1)–(4.4). Fig. 8 shows an example of this process. An examples of the whole execution process of $[q_i, j, -, q_k]$ is shown in Fig. 9. The rules (1), (2), (5), and (7.1)–(7.3) are also used besides (4.1)–(4.4).

(c) Execution of a no-operation $[q_i, j, 0, q_k]$:
A no-operation simply changes the state of M. The rules (1), (2), and (8.1)–(8.2) are used for this operation. Fig. 10 shows an example of the execution process of $[q_i, j, 0, q_k]$.

(d) Execution of a test-if-zero/positive operations $[q_i, j, Z, q_k]$ and $[q_i, j, P, q_\ell]$:
The operations $[q_i, j, Z, q_k]$ and $[q_i, j, P, q_\ell]$ are performed by the rules (9.1)–(9.3), and (9.4)–(9.5), respectively (the rules (1) and (2) are also used). Fig. 11 and Fig. 12 show examples of the execution processes of $[q_i, j, Z, q_k]$ and $[q_i, j, P, q_\ell]$, respectively. Note that which rule group (9.1)–(9.3) or (9.4)–(9.5) is used is determined whether the center part of cell 0 contains the term 2^j.

(e) Rules for the initial/final signal:
The rules (10.1)–(10.5) are the ones for the initial/final signal. When M halts in the final state, the signal "1" is generated by the rule (10.5), and this signal travels leftward indefinitely by the rule (10.2). Note that these rules are not necessary for the simulation itself. But, by them, the contents of the counters (i.e., the final result) are kept unchanged even after the computation of M terminates. Symmetrically to this, by the rules (10.1), (10.3), and (10.4), we can go backward before the initial computational configuration of M.

By above, we can see that A correctly simulates M step by step. It is easy to verify that each rule conserves the total number between left- and right-hand sides, and hence A is an NC-PCA.

Now, we show that the following statement holds: If M is reversible, so is A. Assume M is reversible. It suffices to show that each of the above rules has a different right-hand side from those of the other rules. First, we can easily verify that rules (1), (2), ($3.x$), ($4.x$), (10.1), (10.2), and (10.5) satisfy this constraint by simply comparing their right-hand sides with other rules. The rules (10.3), and (10.4) are also so. Because the state q_0 does not appear as the forth element of a quadruple in δ, hence the right-hand sides of these rules never matches those of ($6.x$), ($7.x$), ($8.x$), and ($9.x$), as well as the others.

Next, we consider the rules (5), which correspond to the quadruple $[q_i, j, d, q_k] \in \delta$ such that $d \in \{+, -\}$. The state q_i may appear as the fourth element of the other quadruples. But, since

t	0	0	0	$i+10$	$\begin{matrix}(m+16)\\-(i+10)\\-\gamma(q_i)\end{matrix}$	$\gamma(q_i)$	0	0	0	0	2^j	0
$t+1$	0	0	$i+10$	0	$\begin{matrix}(m+16)\\-(i+10)\\-\gamma(q_i)\end{matrix}$	0	0	0	$\gamma(q_i)$	0	2^j	0
$t+2$	0	0	0	$i+10$	$\begin{matrix}(m+16)\\-(i+10)\\-\gamma(q_i)\end{matrix}$	0	0	0	0	$\begin{matrix}\gamma(q_i)\\+2^j\end{matrix}$	0	0
$t+3$	0	0	$i+10$	0	$\begin{matrix}(m+16)\\-(i+10)\\-\gamma(q_i)\end{matrix}$	0	$\gamma(q_i)$	2^j	0	0	0	0
$t+4$	0	0	0	$k+10$	$\begin{matrix}(m+16)\\-(k+10)\\-\gamma(q_k)\end{matrix}$	$\gamma(q_k)$	0	2^j	0	0	0	0

Figure 9: Execution of a decrement operation $[q_i, j, -, q_k]$.

t	0	0	0	$i+10$	$\begin{matrix}(m+16)\\-(i+10)\end{matrix}$	0	0	0	0	0	0	0
$t+1$	0	0	$i+10$	0	$\begin{matrix}(m+16)\\-(i+10)\end{matrix}$	0	0	0	0	0	0	0
$t+2$	0	0	0	$k+10$	$\begin{matrix}(m+16)\\-(k+10)\\-\gamma(q_k)\end{matrix}$	$\gamma(q_k)$	0	0	0	0	0	0

Figure 10: Execution of a no-operation $[q_i, j, 0, q_k]$.

t	0	0	0	$i+10$	$(m+16)$ $-(i+10)$ $+2^j$	0	0	0	0	0	0
$t+1$	0	0	$i+10$	0	$(m+16)$ $-(i+10)$ $+2^j$	0	0	0	0	0	0
$t+2$	0	0	0	$k+10$	$(m+16)$ $-(k+10)$ $-\gamma(q_k)$ $+2^j$	$\gamma(q_k)$	0	0	0	0	0

Figure 11: Execution of a test-if-zero operation $[q_i, j, Z, q_k]$ for the case $n_j = 0$.

t	0	0	0	$i+10$	$(m+16)$ $-(i+10)$	0	0	0	0	0	0
$t+1$	0	0	$i+10$	0	$(m+16)$ $-(i+10)$	0	0	0	0	0	0
$t+2$	0	0	0	$\ell+10$	$(m+16)$ $-(\ell+10)$ $-\gamma(q_\ell)$	$\gamma(q_\ell)$	0	0	0	0	0

Figure 12: Execution of a test-if-positive operation $[q_i, j, P, q_\ell]$ for the case $n_j > 0$.

$\gamma(q_i) > 0$ (because $q_i \in (\mathrm{Inc}_0 \cup \mathrm{Inc}_1 \cup \mathrm{Dec}_0 \cup \mathrm{Dec}_1)$), the right-hand sides of the rules (5) never match those of $(6.x)$, $(7.x)$, $(8.x)$, and $(9.x)$.

We finally verify that rules $(6.x)$, $(7.x)$, $(8.x)$, and $(9.x)$ satisfy the reversibility constraint. Let Inc$'$, Dec$'$, Nop$'$, and Test$'$ be the sets of states of M defined as follows.

$$\begin{aligned}
\mathrm{Inc}' &= \{q_k \mid [q_i, j, +, q_k] \in \delta \text{ for some } j \in \{0,1\}, \text{ and } q_i \in Q\} \\
\mathrm{Dec}' &= \{q_k \mid [q_i, j, -, q_k] \in \delta \text{ for some } j \in \{0,1\}, \text{ and } q_i \in Q\} \\
\mathrm{Nop}' &= \{q_k \mid [q_i, j, 0, q_k] \in \delta \text{ for some } j \in \{0,1\}, \text{ and } q_i \in Q\} \\
\mathrm{Test}' &= \{q_k \mid [q_i, j, s, q_k] \in \delta \text{ for some } j \in \{0,1\}, s \in \{Z, P\}, \text{ and } q_i \in Q\}
\end{aligned}$$

For each $q_k \in (\mathrm{Inc}' \cup \mathrm{Dec}' \cup \mathrm{Nop}')$ there is exactly one quadruple containing q_k as the fourth element, since M is reversible (thus the quadruple is of the form $[q_i, j, d, q_k]$ ($q_i \in Q, j \in \{0,1\}$, $d \in \{-, 0, +\}$)). Hence, the rules in $(6.x)$, $(7.x)$, and $(8.x)$ corresponding to this quadruple have different right-hand sides from the others. Next, for each $q_k \in \mathrm{Test}'$, there are at most two quadruples containing q_k as the fourth element since M is reversible. They are of the form $[q_i, j, s, q_k]$ ($q_i \in Q, j \in \{0,1\}, s \in \{Z, P\}$). If there is only one, the rules in $(9.x)$ corresponding to this quadruple satisfy the reversibility constraint as above. In the case there are two, they must be of the forms $[q_i, j, Z, q_k]$ and $[q_i', j', P, q_k]$, because M is reversible. We can see the rules in (9.1)–(9.3), and (9.4)–(9.5) corresponding to the two rules have mutually different right-hand sides, because the center part of the cell 0 should be different between two cases of the contents of the counter j. They also differs from the other rules.

By above, each rule in g has different right-hand side from the others, and thus we can conclude that if M is reversible, A is also reversible. □

Example 3.1 Consider a deterministic RCM(2) $M_2 = (2, Q, \{Z, P\}, \delta, q_0, q_6)$ having the following quadruples as δ.

$$\begin{array}{llll}
[q_0, & 1, & Z, & q_1] \\
[q_1, & 0, & Z, & q_6] \\
[q_1, & 0, & P, & q_2] \\
[q_2, & 0, & -, & q_3]
\end{array}
\qquad
\begin{array}{llll}
[q_3, & 1, & +, & q_4] \\
[q_4, & 1, & +, & q_5] \\
[q_5, & 1, & P, & q_1]
\end{array}$$

M_2 performs the computation $(q_0, n, 0) \overset{*}{\underset{M_2}{\vdash}} (q_6, 0, 2n)$ for any $n(= 0, 1, \cdots)$. An NC-RPCA A_2 constructed from M_2 by the method given in Theorem 3.1 is as follows.

$$A_2 = (\mathbf{Z}, \mathbf{N}_{25}^3, g_2, (0,0,0))$$

The local function g_2 is defined by the following set of rules, and a simulation process of $(q_0, 2, 0) \overset{12}{\underset{M_2}{\vdash}} (q_6, 0, 4)$ is shown in Fig. 13.

Rules by (1) ($x \in \{0, 1, \cdots, 25\}$) :

$[\ 0,\ x,\ 0\ ] \rightarrow [\ 0,\ x,\ 0\ ]$

Rules by (2) ($y \in \{10, 11, \cdots, 16\}$) :

$[\ 0,\ 0,\ y\ ] \rightarrow [\ 0,\ 0,\ y\ ]$

Rules by (3.1):

$[\ 2,\ 0,\ 0\ ] \rightarrow [\ 0,\ 0,\ 2\ ]$
$[\ 2,\ 2,\ 0\ ] \rightarrow [\ 0,\ 2,\ 2\ ]$
$[\ 6,\ 0,\ 0\ ] \rightarrow [\ 0,\ 0,\ 6\ ]$
$[\ 6,\ 1,\ 0\ ] \rightarrow [\ 0,\ 1,\ 6\ ]$

Rules by (3.2):

$[\ 2,\ 1,\ 0\ ] \rightarrow [\ 0,\ 0,\ 3\ ]$
$[\ 2,\ 3,\ 0\ ] \rightarrow [\ 0,\ 2,\ 3\ ]$
$[\ 6,\ 2,\ 0\ ] \rightarrow [\ 0,\ 0,\ 8\ ]$
$[\ 6,\ 3,\ 0\ ] \rightarrow [\ 0,\ 1,\ 8\ ]$

Rules by (3.3):

$[\ 3,\ 0,\ 0\ ] \rightarrow [\ 2,\ 1,\ 0\ ]$
$[\ 3,\ 2,\ 0\ ] \rightarrow [\ 2,\ 3,\ 0\ ]$
$[\ 8,\ 0,\ 0\ ] \rightarrow [\ 6,\ 2,\ 0\ ]$
$[\ 8,\ 1,\ 0\ ] \rightarrow [\ 6,\ 3,\ 0\ ]$

Rules by (3.4):

$[\ 0,\ 0,\ 2\ ] \rightarrow [\ 2,\ 0,\ 0\ ]$
$[\ 0,\ 2,\ 2\ ] \rightarrow [\ 2,\ 2,\ 0\ ]$
$[\ 0,\ 0,\ 6\ ] \rightarrow [\ 6,\ 0,\ 0\ ]$
$[\ 0,\ 1,\ 6\ ] \rightarrow [\ 6,\ 1,\ 0\ ]$

Rules by (4.1):

$[\ 4,\ 0,\ 0\ ] \rightarrow [\ 0,\ 0,\ 4\ ]$
$[\ 4,\ 2,\ 0\ ] \rightarrow [\ 0,\ 2,\ 4\ ]$
$[\ 7,\ 0,\ 0\ ] \rightarrow [\ 0,\ 0,\ 7\ ]$
$[\ 7,\ 1,\ 0\ ] \rightarrow [\ 0,\ 1,\ 7\ ]$

Rules by (4.2):

$[\ 4,\ 1,\ 0\ ] \rightarrow [\ 5,\ 0,\ 0\ ]$
$[\ 4,\ 3,\ 0\ ] \rightarrow [\ 5,\ 2,\ 0\ ]$
$[\ 7,\ 2,\ 0\ ] \rightarrow [\ 9,\ 0,\ 0\ ]$
$[\ 7,\ 3,\ 0\ ] \rightarrow [\ 9,\ 1,\ 0\ ]$

Rules by (4.3):

$[\ 0,\ 0,\ 5\ ] \rightarrow [\ 4,\ 1,\ 0\ ]$
$[\ 0,\ 2,\ 5\ ] \rightarrow [\ 4,\ 3,\ 0\ ]$
$[\ 0,\ 0,\ 9\ ] \rightarrow [\ 7,\ 2,\ 0\ ]$
$[\ 0,\ 1,\ 9\ ] \rightarrow [\ 7,\ 3,\ 0\ ]$

Rules by (4.4):

$[\ 0,\ 0,\ 4\ ] \rightarrow [\ 4,\ 0,\ 0\ ]$
$[\ 0,\ 2,\ 4\ ] \rightarrow [\ 4,\ 2,\ 0\ ]$
$[\ 0,\ 0,\ 7\ ] \rightarrow [\ 7,\ 0,\ 0\ ]$
$[\ 0,\ 1,\ 7\ ] \rightarrow [\ 7,\ 1,\ 0\ ]$

Rules by (5) for $[q2, 0, -, q3]$:

$[\ 12,\ 7,\ 0\ ] \rightarrow [\ 12,\ 7,\ 0\ ]$
$[\ 12,\ 9,\ 0\ ] \rightarrow [\ 12,\ 9,\ 0\ ]$

Rules by (5) for $[q3, 1, +, q4]$:

$[\ 13,\ 4,\ 0\ ] \rightarrow [\ 13,\ 4,\ 0\ ]$
$[\ 13,\ 5,\ 0\ ] \rightarrow [\ 13,\ 5,\ 0\ ]$

Rules by (5) for $[q4, 1, +, q5]$:

$[\ 14,\ 3,\ 0\ ] \rightarrow [\ 14,\ 3,\ 0\ ]$
$[\ 14,\ 4,\ 0\ ] \rightarrow [\ 14,\ 4,\ 0\ ]$

Rules by (6.1) for $[q3, 1, +, q4]$:

$[\ 13,\ 4,\ 6\ ] \rightarrow [\ 14,\ 3,\ 6\ ]$
$[\ 13,\ 5,\ 6\ ] \rightarrow [\ 14,\ 4,\ 6\ ]$

Rules by (6.1) for $[q4, 1, +, q5]$:

$[\ 14,\ 3,\ 6\ ] \rightarrow [\ 15,\ 8,\ 0\ ]$
$[\ 14,\ 4,\ 6\ ] \rightarrow [\ 15,\ 9,\ 0\ ]$

Rules by (7.3) for $[q2, 0, -, q3]$:

$[\ 12,\ 7,\ 4\ ] \rightarrow [\ 13,\ 4,\ 6\ ]$
$[\ 12,\ 7,\ 5\ ] \rightarrow [\ 13,\ 5,\ 6\ ]$
$[\ 12,\ 9,\ 4\ ] \rightarrow [\ 13,\ 4,\ 8\ ]$
$[\ 12,\ 9,\ 5\ ] \rightarrow [\ 13,\ 5,\ 8\ ]$

Rules by (9.1) for $[q0, 1, Z, q1]$:

$[\ 10,\ 15,\ 0\ ] \rightarrow [\ 11,\ 14,\ 0\ ]$
$[\ 10,\ 16,\ 0\ ] \rightarrow [\ 11,\ 15,\ 0\ ]$

Rules by (9.1) for $[q1, 0, Z, q6]$:

$[\ 11,\ 13,\ 0\ ] \rightarrow [\ 16,\ 8,\ 0\ ]$
$[\ 11,\ 15,\ 0\ ] \rightarrow [\ 16,\ 10,\ 0\ ]$

Rules by (9.4) for $[q1, 0, P, q2]$:

$[\ 11,\ 12,\ 0\ ] \rightarrow [\ 12,\ 7,\ 4\ ]$
$[\ 11,\ 14,\ 0\ ] \rightarrow [\ 12,\ 9,\ 4\ ]$

Rules by (9.4) for $[q5, 1, P, q1]$:

$[\ 15,\ 8,\ 0\ ] \rightarrow [\ 11,\ 12,\ 0\ ]$
$[\ 15,\ 9,\ 0\ ] \rightarrow [\ 11,\ 13,\ 0\ ]$

Rules by (10.1) and (10.2):

$[\ 1,\ 0,\ 0\ ] \rightarrow [\ 0,\ 0,\ 1\ ]$
$[\ 0,\ 0,\ 1\ ] \rightarrow [\ 1,\ 0,\ 0\ ]$

Rules by (10.3):

$[\ 1,\ 22,\ 0\ ] \rightarrow [\ 10,\ 13,\ 0\ ]$
$[\ 1,\ 23,\ 0\ ] \rightarrow [\ 10,\ 14,\ 0\ ]$
$[\ 1,\ 24,\ 0\ ] \rightarrow [\ 10,\ 15,\ 0\ ]$
$[\ 1,\ 25,\ 0\ ] \rightarrow [\ 10,\ 16,\ 0\ ]$

Rules by (10.4):

$[\ 16,\ 7,\ 0\ ] \rightarrow [\ 1,\ 22,\ 0\ ]$
$[\ 16,\ 8,\ 0\ ] \rightarrow [\ 1,\ 23,\ 0\ ]$
$[\ 16,\ 9,\ 0\ ] \rightarrow [\ 1,\ 24,\ 0\ ]$
$[\ 16,\ 10,\ 0\ ] \rightarrow [\ 1,\ 25,\ 0\ ]$

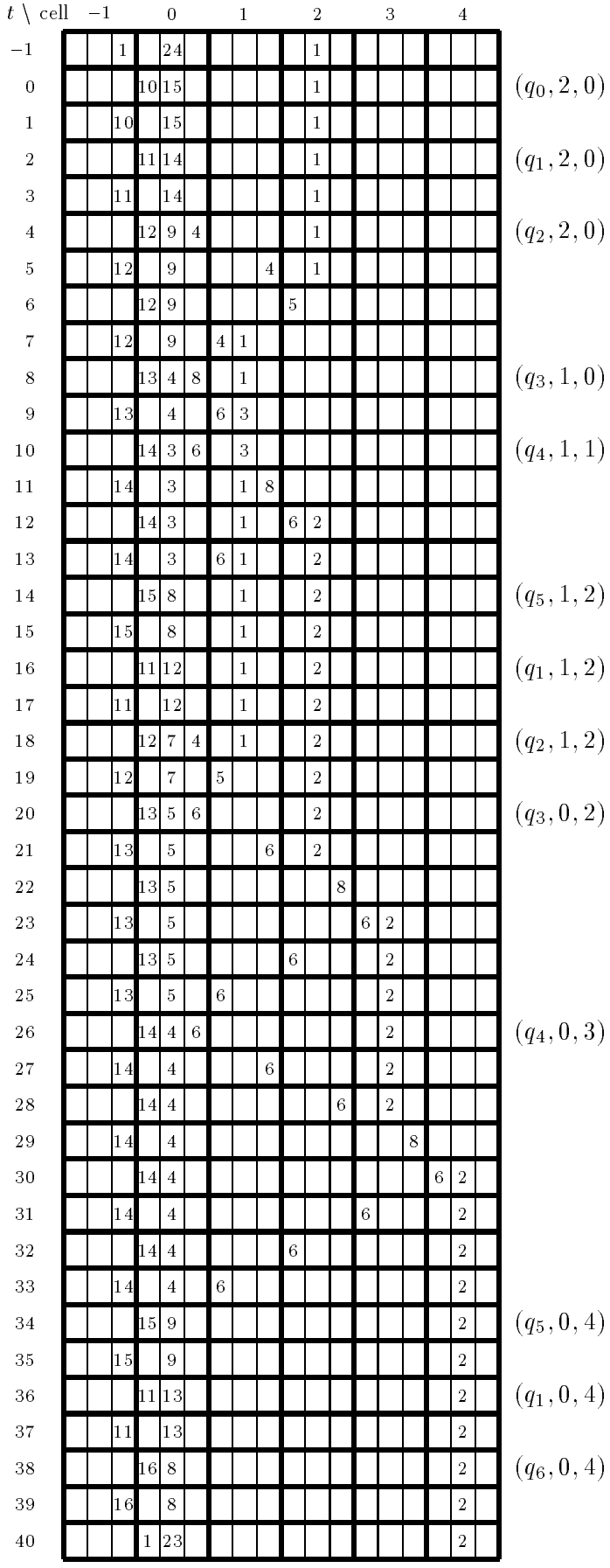


Figure 13: A simulation process of an RCM(2) M_2 by an NC-RPCA A_2.

From Lemma 3.1 and Proposition 3.1, universality of an NC-RPCA is concluded.

Corollary 3.1 An NC-RPCA is computation-universal.

4 Concluding Remarks

In this paper, we proved that an NC-RPCA can simulate a reversible two-counter machine, hence it is computation-universal. In [9], universality of an RCA (not necessarily number-conserving) has been shown by simulating a one-tape reversible Turing machine by an RCA. It is also possible to simulate a one-tape reversible Turing machine by an NC-RPCA. But, if we employ a simulation method in which the contents of each tape square are stored in each cell, then the quiescent state of the NC-RPCA should be $(0, m, 0)$ for some $m > 0$ rather than $(0, 0, 0)$.

References

[1] Albert, J., and Culik II, Karel, A simple universal cellular automaton and its one-way and totalistic version, *Complex Systems*, **1**, 1–16 (1987).

[2] Bennett, C.H., Logical reversibility of computation, *IBM J. Res. Dev.*, **17**, 525–532 (1973).

[3] Bennett, C.H., Notes on the history of reversible computation, *IBM J. Res. Dev.*, **32**, 16–23 (1988).

[4] Fredkin, E., and Toffoli, T., Conservative logic, *Int. J. Theoret. Phys.*, **21**, 219–253 (1982).

[5] Goles, E., Sand pile automata, *Ann. Inst. Henri Poincaré*, **56**, 75–90 (1992).

[6] Goles, E., and Margenstern, M., Sand pile as a universal computer, *Int. J. Modern Physics C*, **7**, 113–122 (1996).

[7] Imai, K., and Morita, K., A computation-universal two-dimensional 8-state triangular reversible cellular automaton, *Proc. of the Second Colloquium on Universal Machines and Computations*, Volume II (Metz), 90–99 (1998).

[8] Margolus, N., Physics-like model of computation, *Physica*, **10D**, 81–95 (1984).

[9] Morita, K., and Harao, M., Computation universality of one-dimensional reversible (injective) cellular automata, *Trans. IEICE Japan*, **E-72**, 758–762 (1989).

[10] Morita, K., and Ueno, S., Computation-universal models of two-dimensional 16-state reversible cellular automata, *IEICE Trans. Inf. & Syst.*, **E75-D**, 141–147 (1992).

[11] Morita, K., Computation-universality of one-dimensional one-way reversible cellular automata, *Inform. Process. Lett.*, **42**, 325–329 (1992).

[12] Morita, K., Universality of a reversible two-counter machine, *Theoret. Comput. Sci.*, **168**, 303–320 (1996).

[13] Toffoli, J., Computation and construction universality of reversible cellular automata, *J. Comput. Syst. Sci.*, **15**, 213–231 (1977).

[14] Toffoli, T., and Margolus, N., Invertible cellular automata: a review, *Physica D*, **45**, 229–253 (1990).

Simulations of graph automata

Codrin Nichitiu & Eric Rémila[1]
LIP, ENS Lyon
46, Allée d'Italie
69364 Lyon Cedex 07, France
Codrin.Nichitiu@ens-lyon.fr & Eric.Remila@ens-lyon.fr

Abstract

We state a definition of the simulation of graph automata, which are machines built by putting copies of the same finite-state automaton at the vertices of a regular graph, reading the states of the neighbors. The graphs considered here are planar, with the elementary circuits of the same length, and form regular tilings of the hyperbolic plane. Thereafter, we present some results of simulation between such graph automata, comparing them to the cellular automata on Cayley graphs, and we conclude with a possible speed hierarchy.

1 Introduction

Since the beginning of the electronic computer era, different machine models have existed, but while the classical sequential computer has been enormously developed, the intrinsic massive parallel models had a harder time to get off, maybe because it is more difficult for us to understand the behavior of such tools and to build and steer them correctly. However, several fundamental questions about the computing power of these models have been successfully solved, showing that namely the cellular automata are at least as powerful as the Turing machines, the classical sequential computer model.

A cellular automaton is a machine built by putting copies of the same finite state automaton (FSA) on the vertices of a "regular" graph. These FSA read their immediate neighbors' states, and switch their state, all at the same moment, according to the state they were in, and according to what they have read. Historically, cellular automata are built on what is called Cayley graphs (Garzon, 1995; Róka, 1994) ; we slightly generalize this notion to what we call graph automata, still built on "regular" graphs.

What is a "regular" graph ? We see below that other interpretations (than Cayley) can be given to this term : for example, a special class of graphs, which are planar, forming regular tilings of a plane (sphere, euclidean or hyperbolic) and which may not be (some of them) Cayley. When considering this class, questions of computing power and speed naturally arise. Here we formalize these questions using the concept of *simulation*, that is instead of building a machine A on a graph G, let us try to build a machine A' on a graph G' such that by looking at its configurations, we can guess what the configurations of A would have been. This is interesting, because when shown possible for A and A' both ways, this means that both are as fast, but maybe with a linear time factor difference, and otherwise, if shown strictly only one way, then one machine is strictly more powerful than the other.

Some of the first results appeared in (Róka, 1994), where Róka proved that all the regular tilings of the euclidean plane are equivalent, i.e. given two tilings, for each machine built on one tiling we can find another one on the other tiling simulating it, that all the trees are also equivalent, and that trees simulate the euclidean planes but the reverse is impossible, because of the different growths of number of neighbors with the distance from any given vertex (polynomial and respectively exponential). Also, several kinds of simulation can be defined, and Bruno Martin, in (Martin, 1998), shows that all the tilings of the hyperbolic plane are equivalent for simulations where a vertex of A is simulated by only one vertex of A', even if one vertex of A' can simulate several vertices of A. However, these simulations are not effective, that is we don't explicitely know which vertex simulates which one, nor how long it takes.

Here we give different simulations, defining them in the section 2 and proving that we can only consider the graph structure when studying the existence of simulations. In section 3 we present a simulation between dual graphs. The result is quite natural, but the construction details have to be treated carefully. In section 4 we give simulations in time 1, using algebraic properties of group theory, our main tool being the notion of Cayley graph. In section 5 we give some corollaries and thereafter we conclude with a possible hierarchy of the "regular" graphs we have considered.

[1]who is also at GRIMA, IUT Roanne (Univ. J. Monnet), 20, ave. de Paris, 42334 Roanne Cedex, France

2 Notion of simulation

We note $a[b]$ the remainder of the euclidean division of two positive integers a by b, and $\pi_q^n(c) = c_q$ where $c = (c_1, c_2, \ldots, c_n)$.

2.1 Graphs

A *graph* G is an ordered pair $G = (X, E)$ where X is a set, and $E \subset X \times X$. The elements of X are called *vertices* and the elements of E *arcs*. A *symmetrical* graph is a graph $G = (X, E)$ such that for each $(x, y) \in E$, it is true that also $(y, x) \in E$. A *planar* graph is (informally) a graph which can be drawn on a sphere so that no arcs cross each-other, and, moreover, it is *locally finite* if there exists a drawing on the plane such that no disc of finite radius of the plane contains an infinite number of vertices of the graph.

The outer and respectively inner *neighborhoods of a vertex* $x \in X$ are the sets $\Gamma_-(x) = \{y \mid (y, x) \in E\}$ and $\Gamma_+(x) = \{y \mid (x, y) \in E\}$, and the respective degrees are $d^-(x) = |\Gamma_-(x)|$ and $d^+(x) = |\Gamma_+(x)|$.

In a symmetrical graph, for all $x \in X$, $d^-(x) = d^+(x) = d(x)$. A *regular* symmetrical graph of degree d is a graph such that for all $x \in X$, $d(x) = d$.

A *chain* in a graph $G = (X, E)$ is a sequence $(x_1, x_2, \ldots, x_n)$, with $x_i \in X$ and $(x_i, x_{i+1}) \in E$. A *circuit* in a graph $G = (X, E)$ is a chain $(x_1, \ldots, x_{n-1}, x_n)$ such that $x_n = x_1$.

An *edge* is a set of two opposed arcs : $\{(x, y), (y, x)\}$, and a *cycle* is a sequence $(x_1, \ldots, x_n, x_1)$ such that $\{(x_i, x_{i+1}), (x_{i+1}, x_i)\}$ are edges. In a symmetrical graph, from any arc we can form an edge, and from any circuit we can form a cycle, by taking its respective opposite. An *elementary cycle* is a cycle such that there is no edge between two non-consecutive vertices in the enumeration of the cycle.

We would like to build what we call *graph automata*, upon such symmetrical planar regular graphs. In order to do that, we label the arcs of the graph, using a function $V_E : E \to 1, 2, \ldots, d$, where d is the degree of the graph. We call then a *labeled graph* an ordered pair $H = ((X, E), V_E)$, where (X, E) is a graph and V_E its labeling function.

An *homogenous* labeling function for a planar labeled graph $G = ((X, E), V_E)$ is a function such that given (informally speaking) a drawing of the graph on a plane, the drawn arcs (x, y) and (x, z) are circularly consecutive in the trigonometric sense (around the drawn vertex x) if and only if $V_E(x, z) = V_E(x, y)[d] + 1$.

We can canonically define, from V_E, in order to ease the notation, the neighborhood functions

$$\begin{array}{lcl} V_X : X \times \{1, 2, \ldots, d\} & \to & X \\ V_R : X \times \{1, 2, \ldots, d\} & \to & \{1, 2, \ldots, d\} \end{array} \quad \text{with} \quad \begin{array}{ll} 1. & V_E(x, V_X(x, i)) = i, \text{ for } 1 \leq i \leq d \text{ and } x \in X \\ 2. & V_R(x, V_E(x, y)) = V_E(y, x) \text{ for } (x, y) \in E \end{array}$$

V_X gives the vertex which is the i-th neighbor of a vertex, and V_R of an arc (given by the origin and the label) gives the label of the opposite arc.

2.2 Graph automata

A *regular graph automaton (GA)* is a triple (H, Q, δ) where $H = ((X, E), V_E)$ is a regular symmetrical labeled graph of degree d, V_E is a labeling function, $Q \subset \mathbb{N}$, $\delta : Q^{d+1} \to Q$ is the transition function of the FSA. A configuration of the graph automata is a function $c : X \to Q$. Given two configurations c_1 and c_2, we note $c_2 = \delta(c_1)$ if

$$c_2(x) = \delta(c_1(x), c_1(x_1), c_1(x_2), \ldots, c_1(x_d)) \qquad \text{with} \quad x_i \in \Gamma_+(x) \text{ such that } V_E(x, x_i) = i$$

Remark : *This actually means that we put copies of the same finite-state automaton in all the vertices of such a graph, having d-tuples of states as input letters.*

2.3 Simulation

Given a positive integer constant T, we say that a GA A' *simulates* a GA A in T time units, if there exists a function ψ such that for each configuration c_1 of A there exists a configuration c_1' of A' such that the diagram from figure 1 be respected.

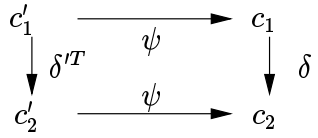


FIG.1

Remark : *We can immediately see that the relation "simulates" is reflexive and transitive. This definition is more general than the one from (Róka, 1994) which is also used by B. Martin in (Martin, 1998) : if one GA simulates another one in the sense of Róka, then it simulates it in our sense, but not conversely.*

One might think the behavior of a GA equally depends on the graph it is built on and on the labeling. However, we prove :

Proposition 1 :
Consider a regular symmetrical graph G. For each pair of its different labelings (V_E, V'_E), for each GA $A = ((G, V_E), Q, \delta)$ there exists a GA $A' = ((G, V'_E), Q', \delta')$, which simulates A in 1 time unit.

Proof

The only difference between two labelings is that for each vertex x there is a permutation t_x, which associates the labels of V_E to the ones of V'_E, locally : $V'_E(x, y) = t_x(V_E(x, y))$. This permutation may not be the same for all the vertices, but there is a finite number $(d!)$ of possible permutations : $t_x : \{1, 2, \ldots, d\} \to \{1, 2, \ldots, d\}$. We construct Q' as a set of $d+1$-tuples : $Q' = Q \times \{1, 2, \ldots, d\}^d$. The permutation of the label i around the vertex x will then be $t_x(i) = \pi_{i+1}^{d+1}(c'(x))$. For $1 \leq i \leq d$, let us denote $\underline{x_i = V_X(x, i)}$, and $\overline{y} = \pi_1^{d+1}(y)$ if y is a $d+1$-tuple. Then δ' is such that $\delta'(c'(x), c'(x_1), \ldots, c'(x_d)) = (\delta(\overline{c'(x)}, \underline{c'(x_{t_x(1)})}, \underline{c'(x_{t_x(2)})}, \ldots, \underline{c'(x_{t_x(d)})}), t_x(1), \ldots, t_x(d))$ and the simulation function ψ is such that $(\psi(c'))(x) = \overline{c'(x)}$ and partially defined, only on valid c' configurations. □

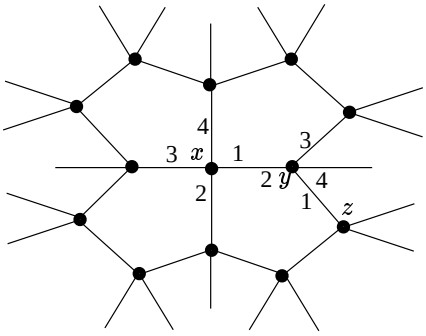
This proposition allows us to extend the definition of simulation to classes of GA's built on the same graphs. We say that a *graph $G = (X, E)$ simulates a graph $G' = (X', E')$* in T time units, if for each V_E, for each GA built on (G, V_E), there exists a V'_E and a GA built on (G', V'_E) which simulates it in T time units. We also see that, since all labelings are equivalent for a simulation, if we consider planar graphs, we can then only consider homogenous labelings.

The graphs we will use to build GA's are of a special type, having all the non-trivial[2] cycles of the same length, and we will first present a simulation between graphs which are dual in the sense of the cycles.

3 Simulations by duality

3.1 $\Gamma[k, d]$

We note with $\Gamma[k, d]$ the (unique up to isomorphism) planar regular symmetrical graph of degree d and locally finite, such as all its non-trivial elementary cycles are of length k. Figure 2 shows an example of numbering of such a graph.
The $\Gamma[k, d]$ graphs possess what we would informally call "regularity". There are other classes of graphs showing types of "regularity" as well, and we will see one in the next section.



$V_X(x, 1) = y$
$V_X(y, 1) = z$
$V_X(y, 2) = x$
$V_E(x, y) = 1$
$V_E(y, x) = 2$
$V_R(x, 1) = 2$
$V_R(y, 2) = 1$

FIG.2 *Example :* $\Gamma[5, 4]$

3.2 Simulation

The graphs[3] $\Gamma[k, d]$ and $\Gamma[d, k]$ are dual in the sense of the elementary cycles (faces) : if in one of the graphs we replace every face by a vertex, and we put an edge between two such new vertices if the two corresponding cycles share an edge, we obtain the other graph. We could ask if there is a simulation between GA's built on these two graphs. The answer is positive :

Theorem 1 (of duality):
For all $k, d \in \mathbb{N}$, $\Gamma[k, d]$ simulates $\Gamma[d, k]$ in $k+1$ time units.

[2] a trivial cycle in a symmetrical graph is (x, y, x) if $(x, y), (y, x) \in E$

[3] k and d are actually such that the graphs are infinite, tiling the euclidean or hyperbolic plane ($\Gamma[3, 3]$, $\Gamma[4, 4]$ and $\Gamma[6, 3]$ are euclidean, and $\Gamma[3, \geq 4]$, $\Gamma[4, \geq 5]$, $\Gamma[5, \geq 4]$, $\Gamma[6, \geq 4]$ and $\Gamma[\geq 7, \geq 3]$ hyperbolic

Proof

Let $A = ((X, E), V_E)$ and $A' = ((X', E'), V'_E)$, with $(X, E) = \Gamma[k, d]$ and $(X', E') = \Gamma[d, k]$.

From a vertex of one of the graphs we would like to obtain the dual cycle, in the other graph. Suppose we number in anti-trigonometric sense and starting from an arbitrary point, the vertices of this dual cycle. One can then consider the families of functions

$$(s_\omega)_{\omega \in X} : \{1, 2, \ldots, d\} \rightarrow X' \qquad \text{and} \qquad (s_x)_{x \in X'} : \{1, 2, \ldots, k\} \rightarrow X$$

such that $(s_\omega(1), s_\omega(2), \ldots, s_\omega(d), s_\omega(1))$ is the dual cycle of ω, in A', $(s_x(1), s_x(2), \ldots, s_x(k), s_x(1))$ is the dual cycle of x, in A, and the arc $(s_\omega(j), s_\omega(j[d]+1))$ of E' is crossed, by duality, by the arc of E labeled by j, that is $(\omega, V_X(\omega, j))$ (see also (Cori, 1984)).

These functions allow us to name the dual cycles for each vertex. Let M' a GA built on A', and let us construct a GA M on A which simulates M'.

Simulation of M' by M We propose a local and finite repartition of the vertices of A' on the ones of A : a vertex ω of A simulates exactly the d vertices $s_\omega(i)$ of the dual cycle in A'.

Remark : *This way, a vertex x of A' will be simulated by k vertices of A, because x belongs (in $A' = (\Gamma[d, k], V'_E)$) to k cycles, to which respectively correspond by duality the k vertices $s_x(1), \ldots s_x(k)$ in A.*

A possibility to realize this simulation consists in the application by M of the same transition of M' $\delta'(c'(x), c'(V'_X(x,1), V'_X(x,2), \ldots, V'_X(x,k)))$ by the vertices of A.

Let us first walk through an example, with $k = 7$. Each vertex simulating x has a stack. Suppose at the initialisation step, these vertices put at the bottom of the stack the state of the central vertex x, and above, the state of the neighboring vertex (on the 'left'). Afterwards, at each step the vertices will copy the top of the stack of their neighbor, and at the end of the period they will know all the necessary states to apply δ'.

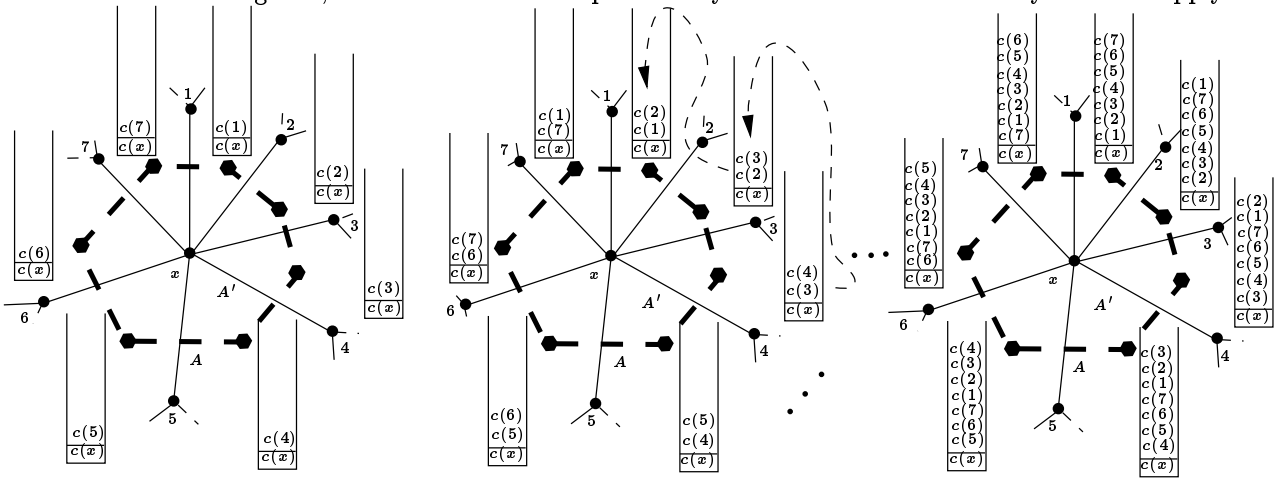


FIG.3 *Evolution of $k = 7$ vertices simulating x, one of their dual vertices*

Therefore in general, since each x in A' receives the informations of k neighbors, to simulate it in A, the vertices have to store the $k+1$-tuples of the form $(c_t(x), c_t(V_X(x,1)), \ldots, c_t(V_X(x,k)))$ (the stacks). On the other hand, a vertex ω of A has to simulate d vertices of A' (its dual cycle), therefore it has d such stacks, and the states of the vertices of A will be the elements of $Q^{d(k+1)}$. This algorithm is then performed in parallel inside each cycle of A.

However, the implementation of this idea brings up three problems. The first has to do with the initialisation of the stack : how does a vertex ω know not only the center x, but also its 'left' neighbor y? Let us remark that this 'left' neighbor belongs to the dual cycle of ω, therefore it is actually simulated as well. Consequently, its state is at the bottom of a another stack of the same ω. Which stack? These stacks are all numbered : the stack j simulates the vertex x if $s_\omega(j) = x$. Then there is also a j' such that the stack j' simulates y, that is $s_\omega(j') = y$. Moreover, y is consecutive to x in the s numbering, and then $j' = j[d] + 1$ (by construction of s). Thus we have solved this problem.

The second problem has to do with the filling of the stacks : since each vertex ω has d stacks, how does such a vertex simulating x know *the number of the stack* of its neighbor which it takes the top from? Let i such that $\omega = s_x(i)$. Then this neighbor of ω is $\omega' = s_x(i[k]+1)$ and the problem is to know the j' such that $x = s_{\omega'}(j')$, i.e. the rank of the stack for x of the neighbor of ω. By definition, s_ω is such that the arc $(s_\omega(j), s_\omega(j[d]+1))$ is crossed, by duality, by the arc $(\omega, V_X(\omega, j))$, labeled by j ; also, $V_E(\omega, \omega') = j[d] + 1$ because V_E is homogenous. Then, the arc (ω', ω) labeled by $j' = V_E(\omega', \omega)$ also crosses, by duality, the arc $(s_{\omega'}(j'), s_{\omega'}(j'[d] + 1))$. Thus $j' = V_R(\omega, j[d] + 1)$, because V_R gives the opposite arc label.

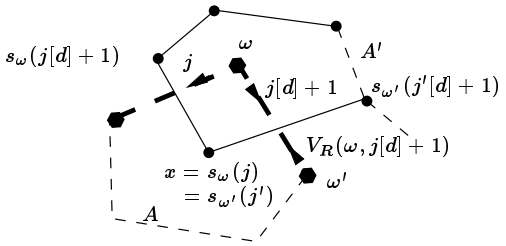


FIG.4 *Dual labeling*

Finally, the third problem appears at the last step, when applying δ'. The stacks are filled with the states of all the neighbors of the vertices to simulate, in the good order, but maybe not starting with the right one (i.e. we would need a circulary permutation to rearrange the content of the stack, see figure 4). We wish to have the state of the vertex $V'_X(x, 1)$ at the bottom of the stack (right above the state of the central vertex, of course). Who is at the bottom? Let us say the state of $V'_X(x, r_i)$ is there, at the bottom of the stack of $\omega = s_x(i)$. Then it is enough to roll the stack upwards $r_i - 1$ times, to be able to apply δ' correctly. We need to find this r_i. This vertex $V'_X(x, r_i)$ has also been called the 'left' neighbor of ω, being actually, for $x = s_\omega(j)$, its successor (in the dual cycle of ω) : the vertex $s_\omega(j[d]+1)$. Here is an example (fig. 5), for $i = 1$:

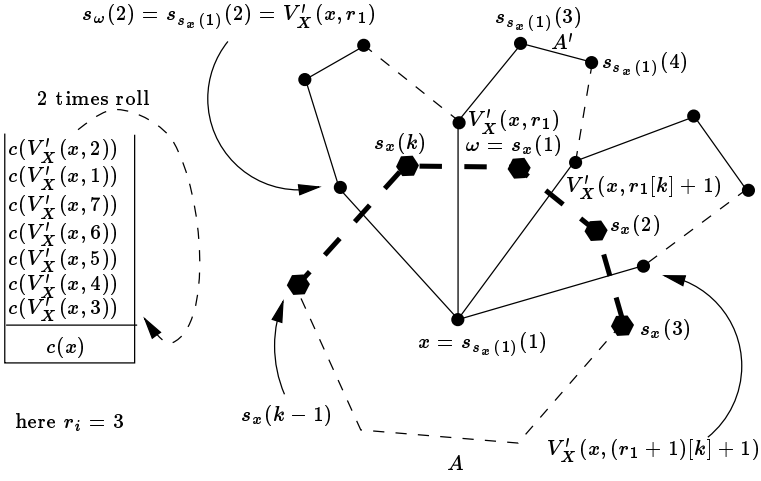


FIG.5 *Simulation of A' (continuous line) by A (dashed line)*

Now, if we replace ω by $s_x(i)$ in the equality $s_\omega(j) = x$, we obtain $x = s_{s_x(i)}(j)$ and then r_i is given by the equality $V'_X(x, r_i) = s_{s_x(i)}(j[d] + 1)$. This way, we have related the numbering of the neighbors of x with that of the dual cycle of ω. The r_i-th neighbor of x is the element which follows x in the dual cycle of $\omega = s_x(i)$. The algorithm thus says that right before the $\omega = s_x(i)$ apply δ', they roll the arguments $r_i - 1$ times. However, these r_i must be given as input ; therefore the states of M will be $d(k+2)$-tuples : d stacks of k elements plus the bottom, and d shifts r_i.

The ψ function is then a projection extracting the bottom of the stacks : for each $x \in A'$, let $\omega \in A$ such that there exists an integer j with $s_\omega(j) = x$. Then

$$c'(x) = (\psi(c))(x) = \text{bottom of the } j\text{-th stack of } \omega (\text{more precisely } \pi^{d(k+2)}_{(j-1)(k+2)+1}(c(\omega)))$$

□

We have said that there are other classes of "regular" graphs, one of them being the class of Cayley graphs. When studying this class, we can benefit from algebraical tools to find conditions for simulations.

4 Simulations induced by morphisms

4.1 Cayley graphs

We see below that there is a non-void intersection of the classes $\Gamma[k, d]$ and Cayley, but that neither is included in the other. In order to define what a Cayley graph is, we need to introduce the notion of presentation.

A *presentation* of a group G is an ordered pair of two sets $\langle A|B\rangle$, $A \in G$, $B \subset A^*$, such that

- for any $\omega \in G$, there exists a word with letters $a_i \in A$ (called *generators*) equal to ω
- all the words of B (called *relators*) are equal to the identity element of the group (1_G)
- any $\omega \in G$ such that $\omega = 1_G$ can be written as $\omega = \prod_{i=1}^{n} u_i w_i u_i^{-1}$ with $u_i \in G$, $w_i \in B$.

A *symmetrical presentation* is a presentation in which for each $a_i \in A$ there exists $a_j \in A$ with $a_i a_j \in B$. Here we only consider symmetrical presentations, which we call presentations. A *homogenous presentation* is a presentation which induces a homogenous V_E.

A *symmetrical Cayley graph* of a group $G = \langle a_1, \ldots, a_d|B\rangle$ is a regular labeled symmetrical graph $H = ((G, E), V_E)$ of degree d such that $(u, w) \in E$ if and only if there exists an integer i with $1 \leq i \leq d$ such that $w = ua_i$. The label $V_E(u, w)$ of (u, w) is therefore i.

Remark : *The circuits of H are labeled with elements of A, according to their order of occurence in the words of B. The function V_R gives, for a vertex x and a generator a_i, the generator labeling the opposite arc, and since if $w = ua_i$, $u = wa_i^{-1}$ and for $a_i \in A$ (as in G) a_i^{-1} is unique, it follows that for a symmetrical Cayley graph, V_R does not depend on the vertex.*

This remark allows us to call $V_R(x, i) = \sigma_d(i)$; therefore for any arc (x, y) of a symmetrical Cayley graph, $V_E(x, y) = \sigma(V_E(y, x))$, and $\sigma(\sigma(i)) = i$.

For example, $(\Gamma[4, 4], V_E)$ is the symmetrical Cayley graph of $G = \langle a, b, c, d|ac, bd, abcd\rangle$, with $V_E(u, ua_i) = i$, for $u \in G$.

4.2 $\Gamma[k, d]$ and symmetrical Cayley graphs

We have said that there is a non-void intersection of the two classes : symmetrical Cayley graphs and $\Gamma[k, d]$ graphs. This problem has been exhaustively studied by T. Chaboud, in (Chaboud, 1995b), and we recall in this subsection his results we need to study the simulations.

Theorem 2 ((Chaboud, 1995b), p. 51) :
$(\Gamma[k, d], V_E)$ is the symmetrical Cayley graph of a group presentation (with the appropriate V_E) if and only if k is divisible by at least an i from $\{2, \ldots, d\}$.

We recall now the way $\Gamma[k, d]$ homogenous presentations look (Chaboud, 1995b). The relators from the presentation encode the circuits of the graph. Here all the elementary non-trivial circuits are of length k, so we have as relators words of k generators. Also, to express the couples $a_i a_i^{-1}$, there are also two-letter words. We have noted with σ_d the permutation such that $a_i = a_{\sigma_d(i)}^{-1}$. Let $p_d(i) = i[d] + 1$.

If the label of an arc of a face of $(\Gamma[k, d], V_E)$ is α, then the next arc to continue the border of the face is the inverse of the generator consecutive to α, i.e. $\sigma_d(p_d(\alpha))$. Then, in order to close the border of the face (in anti-trigonometric sense), we have to walk on k such arcs, which gives the other relators of the presentation, under the form of products

$$\alpha \cdot (\sigma_d \circ p_d)(\alpha) \cdot (\sigma_d \circ p_d)^2(\alpha) \cdot \ldots \cdot (\sigma_d \circ p_d)^{(k-1)}(\alpha)$$

all this for each generator α.

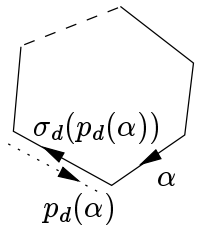


FIG.6 *Border of a face – relators $\sigma_d \circ p_d$*

T. Chaboud shows by induction in that for each Cayley $\Gamma[k, d]$ there exists such a homogenous presentation with d generators : $G = \langle a_1, \ldots, a_d|a_i a_{\sigma_d(i)}, \prod_{j=0}^{k-1} a_{(\sigma_d \circ p_d)^j(i)}\rangle$.

An interesting tool used to represent these presentations is a diagram with d points, a circuit going once through each point (that is p_d) and arcs linking points according to σ_d.

A relator corresponds then to a cycle alternating the arcs : one of p_d, and then one of σ_d, for instance, $(1, d-1, d+1, 3, 2, 1)$ for the figure 4.
Moreover, T. Chaboud shows again by induction that for each Cayley $\Gamma[k,d]$, $k \neq d$ or k not prime, among its homogenous presentations there is one having what Chaboud calls a *handle* , that is : $d = \sigma_d(d-1) = p_d(d-1)$.

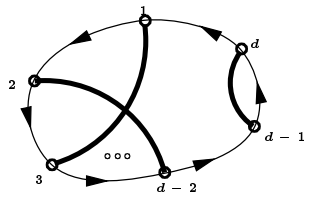


FIG.7 *σ_d-p_d diagram with a handle*

Also, if we have two homogenous presentations with handles for $\Gamma[k,d_1]$ and respectively $\Gamma[k,d_2]$, then, by a fusion operation : $\Gamma[k,d_1] \oplus \Gamma[k,d_2] = \Gamma[k,d_1+d_2-2]$ we can easily find another homogenous presentation for $\Gamma[k,d_1+d_2-2]$.
This operation identifies the vertices of a handle thus creating a new diagram. For example, for the figure 4, if we make the fusion of that diagram with a copy of itself, the new diagram would have $2d$ generators, with d inverse of $2d$, and all the circuits "twice", in the two "compartments" of the new diagram.

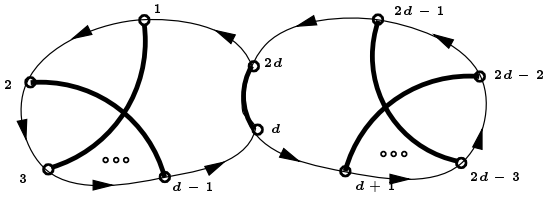


FIG.8 *$\sigma - p$ diagram obtained by fusion*

4.3 Simulations

For a symmetrical Cayley graph C, let us denote with G_C the group presented such that its symmetrical Cayley graph is C.

Based upon a result of Róka from (Róka, 1994), we have found a similar sufficient condition for a simulation to exist between two GA's.

Proposition 2 :

If there exists a (surjective) group morphism ϕ from $G_H = \langle P|Q\rangle$ to $G'_{H'} = \langle P'|Q'\rangle$, such that $P' \subset \phi(P)$, then H simulates H' in 1 time unit.

Proof

Let the GA A' be (H', Q', δ'). We try to build a GA called $A = (H, Q, \delta)$, simulating A'. We set $Q' = Q$. For each configuration c of A such that for any $\omega_1, \omega_2 \in H$, $\phi(\omega_1) = \phi(\omega_2) \Longrightarrow c(\omega_1) = c(\omega_2)$, we define $\psi(c) = c'$ such that for each $\omega \in H$, $c'(\phi(\omega)) = c(\omega)$. c' is well-defined since ϕ is surjective.

Let $d = |P|$, $d' = |P'|$.

Since ϕ is surjective, there exists an injective function $t : \{1, \ldots, d'\} \to \{1, \ldots, d\}$ such that $\phi(a_{t(i)}) = a'_i$. This actually tells us, for an i from 1 to d', which generator of P we must consider so that its image by ϕ be the i-th generator of P'. We then define $\delta(c(\omega), c(\omega a_1), \ldots, c(\omega a_d)) = \delta'(c(\omega), c(\phi(\omega a_{t(1)})), \ldots, c(\phi(\omega a_{t(d')})))$.
□

The reason this proof works is the condition $P' \subset \phi(P)$, which informally insures that the neighborhood of the image $\phi(\omega)$ of a vertex ω be included in the image by ϕ of the neighborhood of ω.

Now we shall look for such morphisms between different $\Gamma[k,d]$ which are symmetrical Cayley graphs, i.e. with k divisible by a number between 2 and d.

We have first

Theorem 3 :

Let $k, d \in \mathbb{N}$. If k is divisible by an integer i with $2 \leq i \leq d$ (i.e. $C = (\Gamma[k,d], V_E)$ is a symmetrical Cayley graph), then for each positive integer n, $\Gamma[nk,d]$ simulates $\Gamma[k,d]$ in 1 time unit.

Proof

Let $A = (\Gamma[k,d], V_E)$, and $A' = (\Gamma[nk,d], V'_E)$. Based upon the presentation of G_A, we can find a

presentation for $G_{A'}$:

$$G_{A'} \quad = \quad \langle a'_1, \ldots, a'_d | a_i a_{\sigma_d(i)}, \left(\prod_{j=0}^{k-1} a'_{(\sigma_d \circ p_d)^j(i)} \right)^n \rangle$$

because this way the circuits become n times longer, while keeping the same number of generators.

Then the function $\phi : G_{A'} \to G_A \quad \phi(a'_i) = a_i \quad \phi(a'_i a'_j) = f_B(a'_i) f_B(a'_j)$ defines very well a morphism, which is surjective, because the relators of the presentations of G_B are very well transported on G_A and we conclude with proposition 2. □

What does the theorem amount to, actually ? There is a "natural" way of mapping the vertices of A to the ones of A' : each circuit $(1, 2, \ldots, k)$ of A is copied n times on a « corresponding » circuit of A' : $(1, 2, \ldots, k, 1, 2, \ldots, k, \ldots, 1, 2, \ldots, k)$. Figure 2 shows an example, for $k = 4$, $d = 5$, $n = 2$. The idea works, because indeed each vertex, when mapped to a vertex in the other graph, sees its neighborhood mapped to the one of its image. The drawback of this simulation is, however, that an infinite number of vertices of A' simulate the same vertex of A.

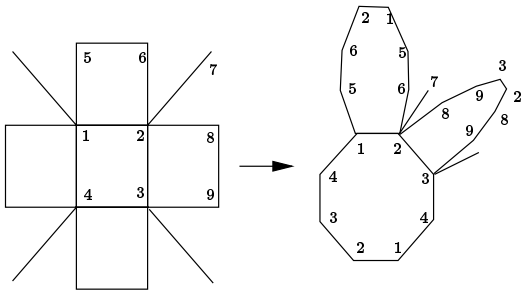


FIG.9 *Simulation by copying the circuits*

Now we would like to modify only d, that is the number of generators. Using a formalism and a lemma from (Chaboud, 1995b), this is made possible.

Theorem 4 (of degree):

Let $k, d \in \mathbb{N}$. If k is divisible by an integer i with $2 \leq i \leq d$ (i.e. $C = (\Gamma[k, d], V_E)$ is a symmetrical Cayley graph), then $\Gamma[k, d+1]$ simulates $\Gamma[k, d]$ in 1 time unit.

Proof

Let $A = (\Gamma[k, d], V_E)$, and $A' = (\Gamma[k, d+1], V'_E)$. For the group $G_{A'}$, keeping the same circuits as the ones of A, we can give as a presentation $G_{A'} = \langle a'_1, \ldots, a'_{d+1} | a'_i a'_{\sigma_{d+1}(i)}, \prod_{j=0}^{k-1} a'_{(\sigma_{d+1} \circ p_{d+1})^j(i)} \rangle$.

Let us apply the handle property to $\Gamma[k, d+1]$, finding a presentation with a handle, for $A' = (\Gamma[k, d+1], V'_E)$. Then we can construct a new one for $\Gamma[k, d]$, namely a permutation σ'_d, which superposes the generators i and j (i.e. d and $d+1$). We obtain in this way a symmetrical Cayley graph $A'' = (\Gamma[k, d], V''_E)$. We have then a morphism $\phi : G_{A'} \to G_A$ such that $\phi(a'_i) = a_i$ for $1 \leq i \leq d$ and $\phi(a'_{d+1}) = a_d$. Therefore, for each GA built on A'' there is one built on A which simulates it, according to proposition 2.

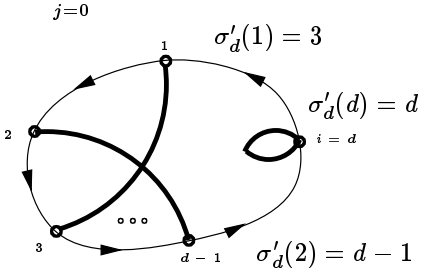


FIG.10 $\sigma - p$ *diagram of* A''

We conclude using the proposition 1, because the only difference between A and A'' is the labeling, so for each GA built on A there is one built on A'' simulating it, and there is another one, built on A, simulating the one on A''. □

Remark : *Geometrically speaking, the presence of a handle in a $\sigma - p$ diagram amounts to the presence of circuits labeled with the same generator a in the symmetrical Cayley graph. Therefore, the theorem simply contracts all these circuits, and manages to conserve the structure.*

We might wish to apply this last idea, maybe slightly modified, to increase k by 1, and not d. We would therefore have for each GA built on $A = (\Gamma[k, d], V_E)$ a GA built on $A' = (\Gamma[k+1, d+c], V'_E)$ simulating it, with $c \geq 0$, thus improving the first theorem about $\Gamma[k, d] \to \Gamma[nk, d]$.

Thinking in terms of relators and graphs, this means we would like to lengthen by one the circuit represented by the words $a_i a_{\sigma(p(i))} \ldots a_{(\sigma \circ p)^{k-1}(i)}$, inserting, for example, a new generator α, which would be its own opposite. This way, the morphism would be well defined, because it would send the a'_i of $G_{A'}$ on the a_i respectively of G_A, and α on 1.

The last problem is 'how big is c?'. the answer is here :

Theorem 5 :
Let $k, d \in \mathbb{N}$, $k \leq d$. If k is divisible by an integer i with $2 \leq i \leq d$ (i.e. $C = (\Gamma[k,d], V_E)$ is a symmetrical Cayley graph), then there exists c, with $1 \leq c \leq \frac{d}{k}$, such that $\Gamma[k+1, d+c]$ simulates $\Gamma[k,d]$ in 1 time unit.

Proof

Let $A = (\Gamma[k,d], V_E)$ and $A' = (\Gamma[k+1, d+c], V'_E)$. We will once again use the Chaboud diagrams representing σ and p. Lengthening a circuit of the diagram (which corresponds to a relator) means, as we said, inserting a loop $\alpha = \sigma(\alpha)$, splitting one of the arcs of p, if there are exactly k taken in the circuit. However, it might be possible that some $\sigma - p$ alternating cycles in the diagrams be of effective length equal to a divisor r of k, and then the insertion of such a loop would increase k by k/r. If $r = 1$ (i.e. for the handles), the problem is solved, because there is no need to insert any loop. Otherwise, we can make sure that $r = k$:

Lemma 1 :
For each $\Gamma[k,d]$ with $k \leq d$ there exists a presentation whose $\sigma - p$ diagram has all its elementary $\sigma - p$ circuits of length k ou 1.

Proof

By induction on the degree d, let $d \in \mathbb{N}$ and suppose the lemma is true for all $d' < d$. Then we can find $d_1 < d > d_2$ with $d_1 + d_2 - 2 = d$; the lemma is true for $\Gamma[k, d_1]$ and respectively for $\Gamma[k, d_2]$, and finally the lemma is also true for $\Gamma[k, d_1 + d_2 - 2] = \Gamma[k,d]$, because the set of all the circuits of the diagram of this last one, obtained by fusion as said in the previous subsection, is simply the union of the sets of circuits of the two combined diagrams. □

We have also to make sure that the circuits are disjoint in the sense of the arcs of p (the ones with arrows), so that the insertion of a loop modifies a well-determined set of circuits. Indeed

Lemma 2 :
For a $\sigma - p$ diagram, if two $\sigma - p$ circuits share the same p arc, then they are identical.

Proof

If x belongs to two circuits and the shared arrow arc starts from x, then $\sigma(p(x))$ also belongs to the two circuits t, and $\sigma(p(\sigma(p(x))))$ too, because they are uniquely determined. We prove as well that all the vertices $(\sigma \circ p)^i(x)$ are common to the two circuits, with $1 \leq i < k$, therefore the circuits are identical. □

Thus, c is equal to the number of such circuits of length k. There is at least one (because $\Gamma[k,k]$ has it, and these circuits are conserved by the fusion operation) and at most d/k because they are disjoint in the sense of the p arcs, which of number d. Finally, the morphism defined as above (the 'copied' generators mapped to their copies, and the 'added' generators, opposite of themselves, mapped to 1) insures the simulation. □

5 Corollaries

As an immediate consequence of this last theorem, we can extend the theorem on $\Gamma[k,d]$ simulated by $\Gamma[k+1, d+c]$ to the case where $k > d$, but with $\Gamma[k,d]$ still symmetrical Cayley graph :

Theorem 6 :
If $k > d$, but k is divisible by a number between 2 and d (i.e. $C = (\Gamma[k,d], V_E)$ is a symmetrical Cayley graph), then $\Gamma[k+1, d]$ simulates $\Gamma[k,d]$.

Proof

Let M be a GA built on $(\Gamma[k,d], V_E)$. According to the theorem of duality, M can be simulated by a GA M^*, built on $(\Gamma[d,k], V_E^*)$. Since $d < k$, $(\Gamma[d,k], V_E^*)$ is a symmetrical Cayley graph, therefore we can apply

the theorem of degree, which says that M^* can be simulated by a GA $M^@$ built on $(\Gamma[d, k+1], V_E^@)$, which, in turn, can be simulated by a GA M' built on $(\Gamma[k+1, d], V_E')$, because of the last theorem once again. □

We have also

Theorem 7 :
For each GA built on $A = (\Gamma[k, d], V_E)$ which is not a symmetrical Cayley graph, there exists GA's respectively built on $A = (\Gamma[k+1, d], V_E')$, on $A'' = (\Gamma[k, nd], V_E'')$, and on $A''' = (\Gamma[k+c, d+1], V_E''')$ which simulate it.

6 Conclusion

We have devised 'ascending' simulations between GA's built on $\Gamma[k, d]$. In general, if $k' \leq k$ and $d' \leq d$, with some special conditions, for a GA built on $(\Gamma[k, d], V_E)$ we can build another one, on $(\Gamma[k', d'], V_E')$ which simulates it in 1 time unit. A special case is the dual one, where the simulation is in time $k+1$ when $\Gamma[k, d]$ simulates $\Gamma[d, k]$.

The morphism-induced simulations belong to a simulation class which may be called *dual-unitary* in an analog sense as the one of Bruno Martin, because a vertex of the simulating GA has to simulate only one vertex of the simulated GA, and his simulation definition is the other way. The first open question coming now is 'what happens downwards', that is 'do there exist dual-unitary simulations of GA's built on $(\Gamma[k', d'], V'E)$ by GA's built on $(\Gamma[k, d], V_E)$?'. We conjecture that this is not possible in time 1.

Then we also can ask 'What if the simulation is also point-to-point, that is dual-unitary and also such that one vertex is simulated by *only one vertex*; can we infer that one of the graphs is a sub-graph of the other, if the simulation is 1 time unit ?' We also conjecture this as true.

Also, more work has to be done to better understand the trade-off between increasing one of (k, d) and decreasing the other, and still keeping possible the simulation, as for the dual graphs.

Finally, a generalisation at GA built on other graphs, still "regular", like the uniform tilings (with a finite family of regular polygons, always in the same order around any vertex) would be of great interest.

References

Blum, M., & Kozen, D. 1978. On the power of the compass (or why mazes are easier to search than graphs). *Pages 132–142 of : Proceedings of the 19th Conference of Foundation of Computer Science.*

Chaboud, T. 1995a. About planar cayley graphs. *In : Proceedings of FCT 1995, LNCS.* Springer Verlag.

Chaboud, T., & Kenyon, C. 1996. Planar cayley graphs with regular dual. *International journal of algebra and computation*, **6**(5), 553–561.

Chaboud, Th. 1995b. *Pavages et graphes de cayley planaires.* Ph.D. thesis, Ecole Normale Supérieure de Lyon. http ://www.ens-lyon.fr/LIP.

Cori, R. 1984. Cartes, hypercartes et leurs groupes d'automorphismes. *In : Séminaire lotharingien.* Univ. Bordeaux 1, UER de Maths et d'Info.

Coxeter, H. S. M., & Moser, W. O. J. 1965. *Generators and relations for discrete groups.* Ergebnisse der Mathematik und ihrer Grenzengebiete - neue Folge, vol. 14. Springer Verlag.

Garzon, M. 1995. *Models of massive parallelism : Analysis of cellular automata and neural networks.* ? Springer Verlag.

Grigorchuk, R. I. 1985. Degrees of growth of finitely generated groups, and the theory of invariant means. *Math. ussr izvestiya*, **25**, 259–300.

Martin, B. 1998. *A geometrical hierarchy of graph automata.* Preprint.

Ramsay, A., & Richtmyer, R. D. 1995. *Introduction to hyperbolic geometry.* Springer Verlag.

Ratcliffe, J. G. 1994. *Foundation of hyperbolic manifolds.* Graduate Texts in Mathematics. Springer Verlag.

Róka, Zs. 1994. *Automates cellulaires sur graphes de cayley.* Ph.D. thesis, Ecole Normale Supérieure de Lyon. http ://www.ens-lyon.fr/LIP.

Is the world a machine?

K. Svozil

Institut für Theoretische Physik, University of Technology Vienna

Wiedner Hauptstraße 8-10/136, A-1040 Vienna, Austria

e-mail: svozil@tph.tuwien.ac.at

www: http://tph.tuwien.ac.at/~svozil

http://tph.tuwien.ac.at/~svozil/publ/mit-svo.tex

Suppose the world is a machine. This is a long-held suspicion, at least as old as the Pythagoreans, that has been revitalized by the early natural sciences. Presently, this intuition is formalized by the computer sciences and constructive as well as discrete mathematics.

Of course, anybody claiming that the world is a machine is in a state of sin, in outright contradiction to the orthodox canon of physics, at least at the moment. We are told that certain quantum mechanical events occur randomly and uncontrollably; and chaos theory pretends that there is randomness even in classical continuum mechanics and electricity.

In principle, the statement that the world is a machine is trivial; a self-fulfilling prophesy if you like. Because anything which can be comprehended can automatically be called machine-like or constructive. Alternatively, if there would be no world comprehension, there would be no talk of the machine-like character of the world. But then there would most probably be no talk at all.

Having said this as a preamble, let me spell out one particular consequence of the assumption that the world is a machine a little bit more explicitly. There has been hardly any feature of quantum mechanics which has given rise to as many fruitless speculations as *complementarity.* Intuitively, complementarity states that it is impossible to (irreversibly) observe certain observables simultaneously with arbitrary accuracy. The more precisely one of these observables is measured, the less precisely can be the measurement of other — complementary — observables. Typical examples of complementary observables are position/momentum

(velocity), angular momentum in the x/y/z direction, and particle number/phase [Per93, WZ83].

The intuition (if intuition makes any sense in the quantum domain) behind this feature is that the act of (irreversible) observation of a physical system causes a loss of information by (irreversibly) interfering with the system. Thereby, the possibility to measure other aspects of the system is destroyed.

As could be expected, this is not the whole story. Indeed, there is reason to believe that — at least up to a certain magnitude of complexity — any measurement can be "undone" by a proper reconstruction of the wave-function. A necessary condition for this to happen is that *all* information about the original measurement is lost. Schrödinger, the creator of wave mechanics, liked to think of the wave function as a sort of *prediction catalog* [Sch35]. This prediction catalogue contains all potential information. Yet, it can be opened only at a *single* particular page. The prediction catalog may be closed before this page is read. Then it could be opened once more at another, complementary, page. By no way it is possible to open the prediction catalog at one page, read and (irreversibly) memorize (measure) the page, and close it; then open it at another, complementary, page. (Two non-complementary pages which correspond to two comeasurable observables can be read simultaneously.)

This may sound a little bit like voodoo. It is tempting to speculate that complementarity can never be modeled by classical metaphors. Yet, classical examples abound. A trivial one is a dark room with a ball moving in it. Suppose that we want to measure its position and its velocity. We first try to measure the ball's position by touching it. This finite contact inevitably causes a finite change of the ball's motion. Therefore, we cannot any longer measure the initial velocity of the ball with arbitrary position.

There are a number of more faithful classical metaphors for quantum complementarity. Take, for instance, Cohen's "firefly-in-a-box" model [Coh89], Wright's generalized urn model [Wri78, Wri90], as well as Aerts' vessel model [Aer82]. In what follows, we are going to explore a model of complementarity pioneered by Moore [Moo56]. It is based on extremely simple systems — probably the simplest systems you can think of : on finite automata. The finite automata we shall consider here are objects which have a finite number of internal states and a finite number of input and output symbols. Their time evolution is mechanistic and can be written down on tables in matrix form. There are no build-in infinities anywhere; no infinite tape or memory, no non-recursive bounds on the runtime *et cetera.*

Let us develop *computational complementarity,* as it is often called [FF83], as a game between two players called Alice and Bob. The rules of the game are as follows. Before the actual game, Alice gives Bob all he needs to know about the intrinsic workings of the automaton. For example, Alice tells Bob, "*if the automaton is in state 1 and you input the symbol 2, then the automaton will make a transition into state 2 and output the symbol 0,*" and so on. Then Alice presents Bob a black box which contains a realization of the automaton. Attached to the black box are two interfaces: a keyboard for the input of symbols, and an output display, on which the output symbols appear. Again, no other interfaces are allowed. In particular, Bob is not allowed to "screw the box open."

Suppose now that Alice chooses some initial state of the automaton. She may either throw a dice, or she may make clever choices using some formalized system. In any case, Alice does not tell Bob about her choice. All Bob has at his disposal are the input-output interfaces.

Bob's goal is to find out which state Alice has chosen. Alice's goal is to fool Bob.

Bob may simply guess or rely on his luck by throwing a dice. But Bob can also perform clever input-output experiments and analyze his data in order to find out. Bob wins if he gives the correct answer. Alice wins if Bob's guess is incorrect. (So, Alice has to be really mean and select worst-case scenarios).

Suppose that Bob tries very hard. Is cleverness sufficient? Will Bob always be able to uniquely determine the initial automaton state?

The answer to that question is "no." The reason is that there may be situations when Bob's input causes an irreversible transition into a black box state which does not allow any further queries about the initial state.

What has been introduced here as a game between Alice and Bob is what the mathematicians have called the *state identification problem* [Moo56, Cha65, Con71, Bra84]: given a finite deterministic automaton, the task is to locate an unknown initial state. Thereby it is assumed that only *a single* automaton copy is available for inspection. That is, no second, identical, example of the automaton can be used for further examination. Alternatively, one may think of it as choosing at random a single automaton from a collection of automata in an ensemble differing only by their initial state. The task then is to find out which was the initial state of the chosen automaton.

The logico-algebraic structure of the state identification problem has been introduced in [Svo93], and subsequently studied in [Svo93, SS94, SS95, SS96, DPS95, SZ96, ST96, CCSY97].

Let us consider an example of the complementarity game. Finite automata, in particular Moore (Mealy) machines are represented by flow tables and state graphs. To illustrate this, assume a Mealy automaton M_s with three states $1,2,3$, three input symbols 1,2,3 and two output symbols 0,1. Input and output symbols are separated by a comma. Arrows indicate transitions. That is, $M_s = (S,I,O,\delta,\lambda)$, with

$$\begin{aligned} S &= \{1,2,3\}, \\ I &= \{1,2,3\}, \\ O &= \{0,1\}. \end{aligned}$$

Its transition and output functions are ($\delta_{s,x}$ stands for the Kronecker delta function)

$$\begin{aligned} \delta(s,i) &= i, \\ \lambda(s,i) &= \delta_{s,i} = \begin{cases} 1 & \text{if } s = i \\ 0 & \text{if } s \neq i \end{cases} \end{aligned}$$

The flow table and state graph of this Mealy automaton is given in Fig. 1.

To illustrate the construction of the automaton propositional calculus, consider the Mealy automaton M_s discussed above. Input/output experiments can be performed by the input of just one symbol i (in this example, more inputs yield no finer partitions). Suppose again that Bob does not know the automaton's initial state. So, Bob has to choose between the input of symbols 1,2, or 3. If Bob inputs, say, symbol 1, then he obtains a definite answer whether the automaton was in state 1 — corresponding to output 1; or whether the automaton was not in state 1 — corresponding to output 0. The latter proposition "not 1" can be identified with the proposition that the automaton was either in state 2 or in state 3.

Likewise, if Bob inputs symbol 2, he obtains a definite answer whether the automaton was in state 2 — corresponding to output 1; or whether the automaton was not in state 2 — corresponding to output 0. The latter proposition "not 2" can be identified with the proposition that the automaton was either in state 1 or in state 3. Finally, if Bob inputs symbol 3, he obtains a definite answer whether the automaton was in state 3 — corresponding to output 1; or whether the automaton was not in state 3 — corresponding to output 0. The latter proposition "not 3" can be identified with the proposition that the automaton was either in state 1 or in state 2.

Recall that Bob can actually perform only one of these input-output experiments. This experiment will irreversibly destroy the initial automaton state (with

$M_s =$

S\I	1	2	3	1	2	3
1	1	1	1	1	0	0
2	2	2	2	0	1	0
3	3	3	3	0	0	1

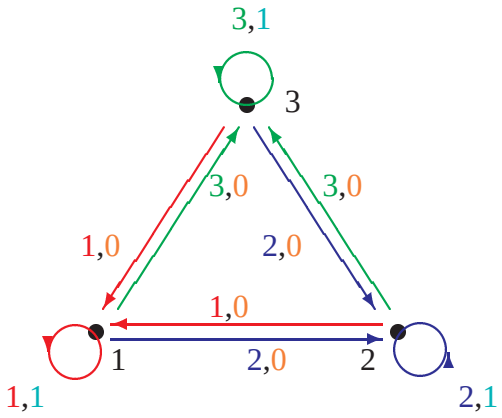


Figure 1: Simplest Mealy automaton featuring computational complementarity.

the exception of a "hit"; i.e., of output 1). Let us thus describe the three possible types of experiment as follows.

- Bob inputs the symbol 1.
- Bob inputs the symbol 2.
- Bob inputs the symbol 3.

The corresponding observable propositions are:

$p_{\{1\}} \equiv \{1\}$: On input 1, Bob receives the output symbol 1.

$p_{\{2,3\}} \equiv \{2,3\}$: On input 1, Bob receives the output symbol 0.

$p_{\{2\}} \equiv \{2\}$: On input 2, Bob receives the output symbol 1.

$p_{\{1,3\}} \equiv \{1,3\}$: On input 2, Bob receives the output symbol 0.

$p_{\{3\}} \equiv \{3\}$: On input 3, Bob receives the output symbol 1.

$p_{\{1,2\}} \equiv \{1,2\}$: On input 3, Bob receives the output symbol 0.

Note that, in particular, $p_{\{1\}}, p_{\{2\}}, p_{\{3\}}$ are not comeasurable. Note also that, for $\varepsilon_{ijk} \neq 0$, $p'_{\{i\}} = p_{\{j,k\}}$ and $p_{\{j,k\}} = p'_{\{i\}}$; or equivalently $\{i\}' = \{j,k\}$ and $\{j,k\} = \{i\}'$.

In that way, we naturally arrive at the notion of a *partitioning* of automaton states according to the information obtained from input/output experiments. Every element of the partition stands for the proposition that the automaton is in (one of) the state(s) contained in that partition. Every partition corresponds to a quasi-classical Boolean block. Let us denote by $v(x)$ the block corresponding to input (sequence) x. Then we obtain

no input:

$$v(\emptyset) = \{\{1,2,3\}\},$$

one input symbol:

input		output 1		output 0
$v(1)$	$=$	$\{\{1\}$	,	$\{2,3\}\}$
$v(2)$	$=$	$\{\{2\}$	,	$\{1,3\}\}$
$v(3)$	$=$	$\{\{3\}$	,	$\{1,2\}\}$.

Conventionally, only the finest partitions are included into the set of state partitions.

Just as in quantum logic, the *automaton propositional calculus* and the associated *partition logic* is the *pasting* of all the blocks of partitions $v(i)$ on the atomic level.

For the Mealy automaton M_s discussed above, the pasting renders just the horizontal sum — only the least and greatest elements $0, 1$ of each 2^2 is identified— and one obtains a "Chinese lantern" lattice MO_3. The Hasse diagram of the propositional calculus is drawn in Figure 2.

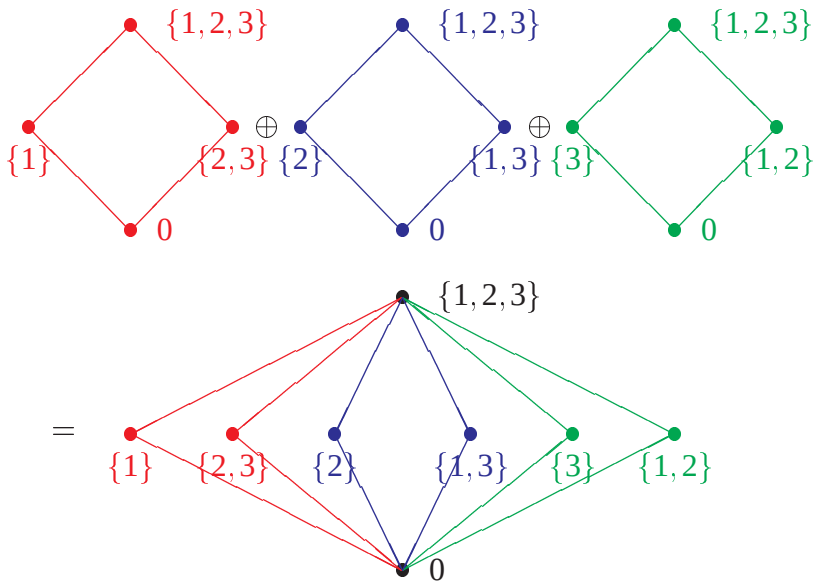


Figure 2: Hasse diagram of the propositional calculus of the Mealy automaton drawn in Figure 1.

Let us go still a little bit further and ask which automaton games of the above kind can people play. This requires the systematic investigation of all possible non-isomorphic automaton propositional structures, or, equivalently, partition logics [Svo93, SS94, SS95, SS96]. In Fig. 3, the Hasse diagrams of all nonisomorphic four-state automaton propositional calculi are drawn.

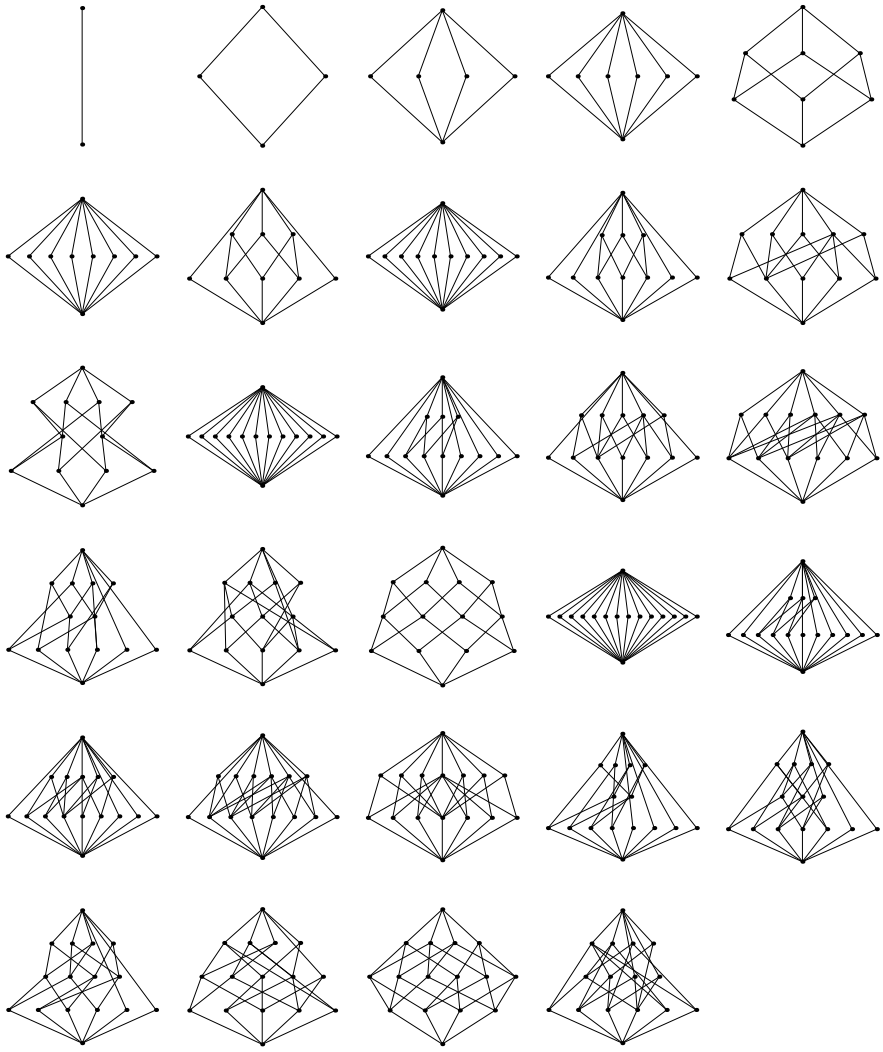


Figure 3: Variations of the complementarity game up to four automaton states.

New automata can be composed from old ones by parallel and serial compositions. In Figures 4 and 5, the Hasse diagrams for simple parallel compositions of two and three automata are drawn [Svo98].

Recall that the method introduced here is not directly related to diagonalization and is a second, independent source of undecidability. It is already realizable at an elementary pre-diagonalization level, i.e., without the requirement of computational universality or its arithmetic equivalent. The corresponding machine model is the class of finite automata.

Since any finite state automaton can be simulated by a universal computer, complementarity is a feature of sufficiently complex deterministic universes as well. To put it pointedly: if the physical universe is conceived as the product of a universal computation, then complementarity is an inevitable and necessary feature of the perception of intrinsic observers. It cannot be avoided.

Conversely, any computation can be realized by a sufficiently complex finite automaton. Therefore, the class of all complementary games is a unique one, encompassing all possible deterministic universes.

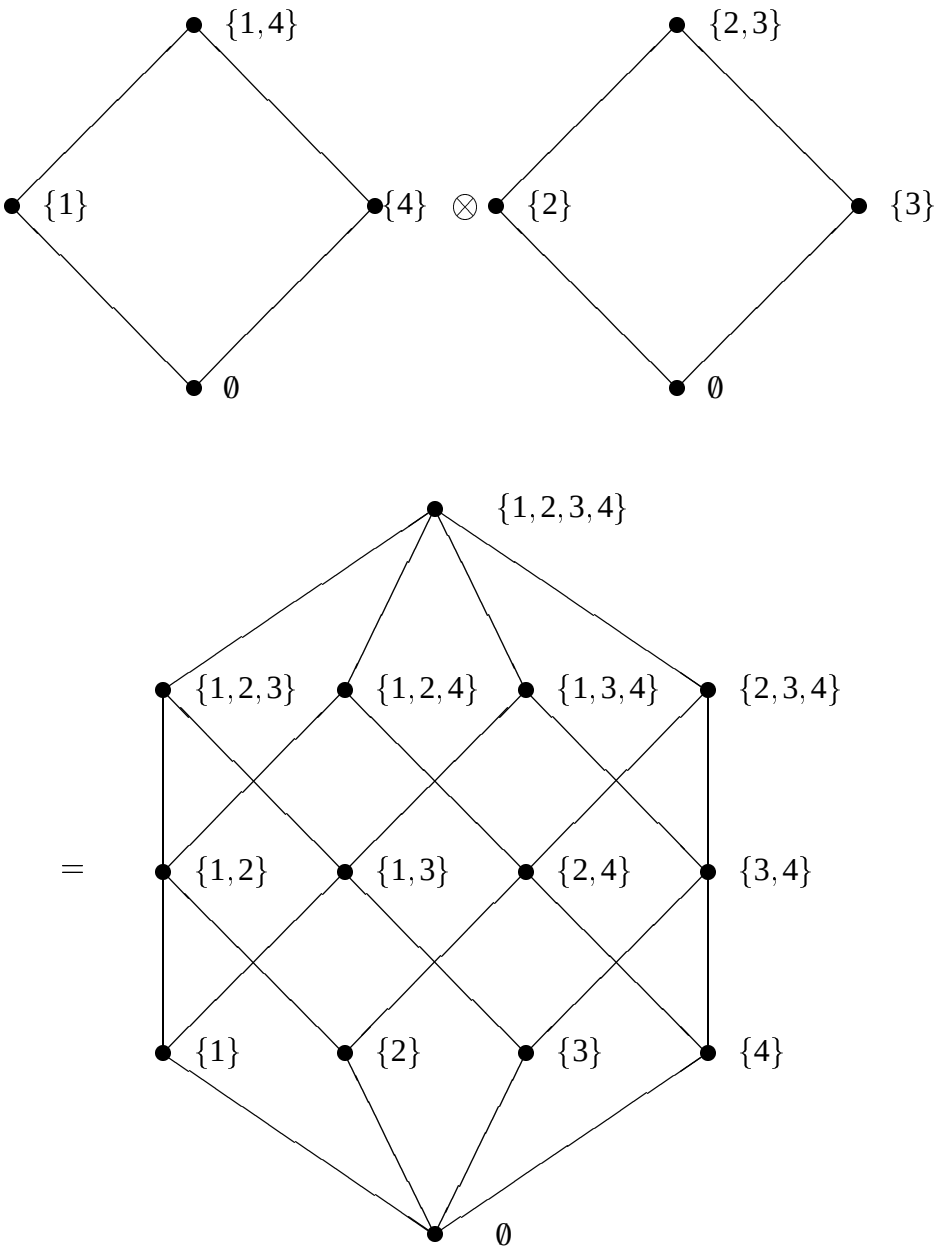


Figure 4: Hasse diagram of the automaton resulting from a parallel composition of two automata.

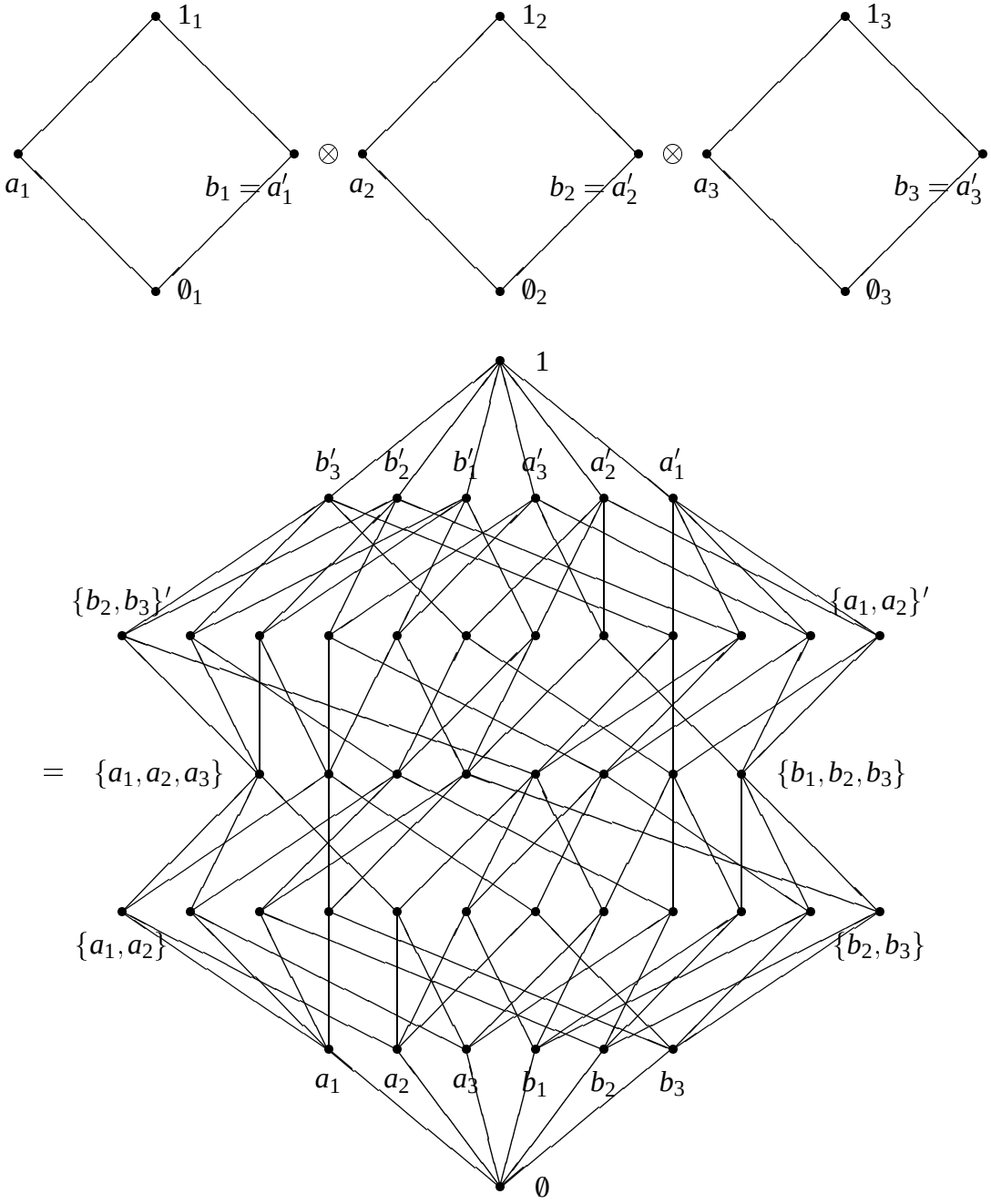


Figure 5: Hasse diagram of the automaton resulting from a parallel composition of three automata.

References

[Aer82] Aerts, D., *Example of a macroscopic classical situation that violates Bell inequalities*, Lettere al Nuovo Cimento **34** (1982), no. 4, 107–111.

[Bra84] Brauer, W., *Automatentheorie*, Teubner, Stuttgart, 1984.

[CCSY97] Calude, Cristian, Calude, Elena, Svozil, Karl, and Yu, Sheng, *Physical versus computational complementarity I*, International Journal of Theoretical Physics **36** (1997), no. 7, 1495–1523.

[Cha65] Chaitin, Gregory J., *An improvement on a theorem by E. F. Moore*, IEEE Transactions on Electronic Computers **EC-14** (1965), 466–467.

[Coh89] Cohen, D. W., *An introduction to hilbert space and quantum logic*, Springer, New York, 1989.

[Con71] Conway, J. H., *Regular algebra and finite machines*, Chapman and Hall Ltd., London, 1971.

[DPS95] Dvurečenskij, Anatolij, Pulmannová, Sylvia, and Svozil, Karl, *Partition logics, orthoalgebras and automata*, Helvetica Physica Acta **68** (1995), 407–428.

[FF83] Finkelstein, David and Finkelstein, Shlomit R., *Computational complementarity*, International Journal of Theoretical Physics **22** (1983), no. 8, 753–779.

[Moo56] Moore, Edward F., *Gedanken-experiments on sequential machines*, Automata Studies (Princeton) (Shannon, C. E. and McCarthy, J., eds.), Princeton University Press, Princeton, 1956.

[Per93] Peres, Asher, *Quantum theory: Concepts and methods*, Kluwer Academic Publishers, Dordrecht, 1993.

[Sch35] Schrödinger, Erwin, *Die gegenwärtige Situation in der Quantenmechanik*, Naturwissenschaften **23** (1935), 807–812, 823–828, 844–849, English translation in [Tri80] and [WZ83, pp. 152-167].

[SS94] Schaller, Martin and Svozil, Karl, *Partition logics of automata*, Il Nuovo Cimento **109B** (1994), 167–176.

[SS95] Schaller, Martin and Svozil, Karl, *Automaton partition logic versus quantum logic*, International Journal of Theoretical Physics **34** (1995), no. 8, 1741–1750.

[SS96] Schaller, Martin and Svozil, Karl, *Automaton logic*, International Journal of Theoretical Physics **35** (1996), no. 5, 911–940.

[ST96] Svozil, Karl and Tkadlec, Josef, *Greechie diagrams, nonexistence of measures in quantum logics and Kochen–Specker type constructions*, Journal of Mathematical Physics **37** (1996), no. 11, 5380–5401.

[Svo93] Svozil, Karl, *Randomness & undecidability in physics*, World Scientific, Singapore, 1993.

[Svo98] Svozil, K., *Quantum logic*, Springer, Singapore, 1998.

[SZ96] Svozil, Karl and Zapatrin, Roman R., *Empirical logic of finite automata: microstatements versus macrostatements*, International Journal of Theoretical Physics **35** (1996), no. 7, 1541–1548.

[Tri80] Trimmer, J. D., *The present situation in quantum mechanics: a translation of Schrödinger's "cat paradox"*, Proc. Am. Phil. Soc. **124** (1980), 323–338, Reprinted in [WZ83, pp. 152-167].

[Wri78] Wright, Ron, *The state of the pentagon. A nonclassical example*, Mathematical Foundations of Quantum Theory (New York) (Marlow, A. R., ed.), Academic Press, New York, 1978, pp. 255–274.

[Wri90] Wright, Ron, *Generalized urn models*, Foundations of Physics **20** (1990), 881–903.

[WZ83] Wheeler, John Archibald and Zurek, Wojciech Hubert, *Quantum theory and measurement*, Princeton University Press, Princeton, 1983.

Cellular Algorithms with 1-Bit Inter-Cell Communications

Hiroshi Umeo
Osaka Electro-Communication University
Faculty of Information Engineering, Department of Informatics
Neyagawa-shi, hatsu-cho, 18-8, Osaka, 572, Japan
e-mail: umeo@umeolab.osakac.ac.jp

Abstract

We propose several cellular algorithms for a special class of cellular automata, CA_{1-bit}, whose inter-cell communication is restricted to 1-bit. The problems we consider are real-time generation of non-regular binary sequences, firing squad synchronization problem and connectivity recognition problem for two-dimensional binary images. It is shown that infinite Fibonacci sequences can be generated in real-time by one-dimensional CA_{1-bit}. In addition, we propose linear- and optimum-time firing squad synchronization algorithms with 1-bit inter-cell communications and, in the last, we show that a set of two-dimensional binary connected images of size $m \times n$ con be recognized in exactly $2(m+n)$ steps by two-dimensional CA_{1-bit}.

The full version of this paper is not available as a postscript file.

Simulations between alternating CA, alternating TM and circuit families

Frank Reischle and Thomas Worsch
Fakultät für Informatik, Universität Karlsruhe

Abstract

Variants of cellular automata consisting of alternating instead of deterministic finite automata are investigated, so-called uniform alternating CA (ACA) and two types of nonuniform ACA. The former two have been considered by Matamala (1997). It is shown that the nonuniform ACA are time equivalent. The main contributions are fast simulations of ACA by uniform circuit families and vice versa. It is shown that nonuniform ACA are time equivalent to circuit families with unbounded fan-in, and that uniform ACA are time equivalent to circuit families with constant fan-in. Hence uniform ACA and alternating TM are time equivalent, too, solving a problem left open by Matamala. The results also give some evidence that a time simulation of nonuniform ACA by ATM is "unlikely" to exist.

1 Introduction

The standard model of deterministic cellular automata (CA) has been generalized in several directions, e.g. nondeterministic CA or stochastic CA. Recently they have also been extended using the concept of alternation [1]. First results have been obtained by Krithivasan and Mahajan [2]. They are considering a rather restricted model, one would call a special type of uniform ACA (UACA), using the terminology of Matamala [3].

Besides uniform ACA he considers so-called weak ACA (WACA) and one variant of nonuniform ACA (denoted $\exists\forall$–ACA in this paper). In [3] it is shown that UACA and WACA are time equivalent (i.e. they can simulate each other with only constant slowdown), that these models can simulate alternating Turing machines (ATM) with constant slowdown, and that these models can be simulated by $\exists\forall$–ACA with constant slowdown.

The rest of the paper is organized as follows: In Section 2 some basic notation is introduced as well as some concepts which are common to all alternating devices. In Section 3 two variants of nonuniform ACA ($\exists\forall$–ACA and $\forall\exists$–ACA) are considered. We prove that these models can simulate each other with only constant slowdown. Section 4 is devoted to uniform ACA (UACA), weak ACA (WACA) and alternating TM (ATM). We show that they are *all* time equivalent, solving a problem left open in [3]. In Section 5 relations between uniform resp. nonuniform ACA and circuit families with bounded fan-in resp. unbounded fan-in are investigated. Therefore the question whether nonuniform ACA are time equivalent to ATM is related to the corresponding

problem for circuit families with (un-)bounded fan-in, giving some indication that nonuniform ACA may be somewhat more powerful than ATM.

2 Basic notions

In this paper we are only dealing with one-dimensional cellular automata (CA) with von Neumann neighborhood N of radius 1. Thus a deterministic CA is specified by a *set of states* Q and a *local transition function*[1] $\delta : Q^N \to Q$ specifying for each *local configuration* $l : N \to Q$ the new state $\delta(l)$ of the central cell. A *(global) configuration* is a mapping $c : \mathbf{Z} \to Q$. The state of cell i in c is denoted c_i and the local configuration "observed" by cell i in its neighborhood is denoted $c_{i+N} : N \to Q : n \mapsto c_{i+n}$. For a configuration c denote by c^N the mapping $c^N : \mathbf{Z} \to Q^N : i \mapsto c_{i+N}$. For a subset $T \subseteq \mathbf{Z}$ we write $c \vdash_{T,\delta} c'$ iff $i \in T \Longrightarrow c'_i = \delta(c_i^N)$ (where we have written c_i^N instead of $c^N(i)$). The relation $c \vdash_{\mathbf{Z},\delta} c'$ is abbreviated to $c \vdash_\delta c'$ and describes one step of a deterministic CA according to the local rule δ.

Of course the usual definition of deterministic CA is captured by the above seemingly unnecessarily complicated formalism. The reason for introducing it is, that it will turn out to be useful for the definition of alternating CA.

For all types of alternating CA it is assumed that the set of states $Q = Q_\exists \cup Q_\forall$ is partitioned into *existential states* $q \in Q_\exists$ and *universal states* $q \in Q_\forall$. The *local transition function* is now of the form $\nu : Q^N \to 2^Q$, specifying a *subset* $\delta(l)$ of possible new states for a cell observing l in its neighborhood. A configuration is defined as for deterministic CA.

For the recognition of formal languages an *input alphabet* $A \subset Q$ and a set $F \subset Q$ of *accepting final states* has to be specified, as well as a *quiescent state* $\Box$. For the quiescent local configuration $l_\Box : n \mapsto \Box$ the local function has to satisfy $\nu(l_\Box) = \{\Box\}$. In the *initial configuration* c_w for an input $w = x_1 \cdots x_n \in A^+$ of length $n \geq 1$ cell i holds input symbol x_i (for $1 \leq i \leq n$) and all other cells are in state $\Box$. A configuration c is *accepting* iff $c_1 \in F$.

The recognition of formal languages by alternating devices is usually defined using the notion of a *computation tree*. Its nodes are configurations and the root is always an initial configuration. In general there are several computation trees with the same root. A computation tree is accepting iff all of its leaves are accepting configurations. A word is accepted if there is an accepting computation tree with c_w as root.

Two machines (be it CA, TM, ...) are called *equivalent* iff they recognize the same language. They are called *time equivalent* iff their time[2] complexities only differ by a constant factor.

The main differences between the various types of alternating CA lie in the definitions of which trees are legal computation trees. This topic is treated separately in the following sections.

[1] B^A denotes the set of all mappings from A to B.

[2] For ACA and TM this means the number of steps, for circuit families their depth.

3 Nonuniform alternating CA

First we define two types of *nonuniform cellular automata*, denoted as $\exists\forall$–ACA and $\forall\exists$–ACA. While $\exists\forall$–ACA are considered in [3] the $\forall\exists$–ACA are equally natural to look at.

For a configuration c let $\exists(c) = \{i \in \mathbf{Z} \mid c_i \in Q_\exists\}$ and $\forall(c) = \{i \in \mathbf{Z} \mid c_i \in Q_\forall\}$. Let $\bar{c}$ and $\bar{c}'$ be two mappings $\mathbf{Z} \to Q \cup Q^N$ and $T \subset \mathbf{Z}$ a subset of cells. We write $\bar{c} \vdash_{T,\nu} \bar{c}'$ iff $i \notin T \Longrightarrow \bar{c}_i = \bar{c}'_i$ and $i \in T \Longrightarrow \bar{c}_i \in Q^N \wedge \bar{c}'_i \in \nu(\bar{c}_i)$.

Using subsets $\bar{C}$, $\bar{C}'$, existential and universal "substeps" are now defined as follows: $\bar{c} \vDash^\forall_T \bar{C}'$ iff $\bar{C}' = \{\bar{c}' \mid \bar{c} \vdash_{T,\nu} \bar{c}'\}$ and $\bar{C} \vDash^\exists_T \bar{C}'$ iff $|\bar{C}| = |\bar{C}'|$ and for each $\bar{c} \in \bar{C}$ there is a $\bar{c}' \in \bar{C}'$ such that $\bar{c} \vdash_{T,\nu} \bar{c}'$.

The one-step relation for $\exists\forall$–ACA is now defined as $c \vDash^{\exists\forall} C'$ iff there is a $\bar{c}$, such that[3] $c^N \vDash^\exists_{\exists(c)} \bar{c} \vDash^\forall_{\forall(c)} C'$. For $\forall\exists$–ACA we use $c \vDash^{\forall\exists} C$ iff there is a set $\bar{C}$, such that $c^N \vDash^\forall_{\forall(c)} \bar{C} \vDash^\exists_{\exists(c)} C'$. One should note that both substeps always refer to the original configuration c.

A tree of configurations is a computation tree for an $\exists\forall$–ACA iff for each non-leaf node c the set $\mathrm{Succ}(c)$ of its successors in the tree satisfies $c \vDash^{\exists\forall} \mathrm{Succ}(c)$. Analogously for $\forall\exists$–ACA all non-leaves c have to satisfy $c \vDash^{\forall\exists} \mathrm{Succ}(c)$.

Lemma 3.1 *For each $\exists\forall$–ACA there is a time equivalent one such that all configurations occuring in any computation tree are such that either all non-quiescent states are existential or they are all universal. The same holds for $\forall\exists$–ACA.*

Proof. The proof is shown only for $\exists\forall$–ACA with von Neumann neighborhood $N = \{-1, 0, 1\}$. (The proofs for other N and $\forall\exists$–ACA can be done analogously). Let Q denote the set of states of an $\exists\forall$–ACA M. We construct another $\exists\forall$–ACA M' with state set Q', the same input alphabet A and the same quiescent state $\square$, which uses one full step to simulate an existential substep and a subsequent full step to simulate the corresponding universal substep of a step of M.

The simulation of an existential substep (resp. a universal substep) is indicated by a special flag ("E" resp. "U") which is stored additionally in each cell of M'. The state set of M' is $Q' = A \cup \{\square\} \cup (\{\mathrm{E}, \mathrm{U}\} \times Q) \cup (\{\mathrm{U}\} \times Q^3)$.

For $x, y, z \in Q$ the states $(\mathrm{U}, y), (\mathrm{U}, xyz) \in Q'$ are universal. States $(\mathrm{E}, y) \in Q'$ are existential as well as all states in A, because the first step of M' will be an existential one. By convention $\square$ is existential, too (although it doesn't matter). The set of accepting final states for M' is $F' = F \cup (\{\mathrm{E}\} \times F)$, where F is the set of accepting final states for M. The transition function ν' of M' is defined in terms of ν of M as follows. For $x', y', z' \in Q'$, and using $\nu'(x', y', z')$ as a shorthand notation[4] we have:

$$\begin{aligned}
\nu'((\mathrm{E}, x), (\mathrm{E}, y), (\mathrm{E}, z)) &= \begin{cases} \{(\mathrm{U}, \hat{y}) \mid \hat{y} \in \nu(x, y, z)\} & \text{if } y \in Q_\exists \\ \{(\mathrm{U}, xyz)\} & \text{if } y \in Q_\forall \end{cases} \\
\nu'(x', (\mathrm{U}, y), z') &= \{(\mathrm{E}, y)\} \\
\nu'(x', (\mathrm{U}, xyz), z') &= \{(\mathrm{E}, \hat{y}) \mid \hat{y} \in \nu(x, y, z)\}
\end{aligned}$$

Furthermore a cell in state $\square$ having a neighbor with flag E (resp. U) acts like a cell in state $(\mathrm{E}, \square)$ (resp. $(\mathrm{U}, \square)$). A cell in state $x \in A$ acts like a cell in state (E, x). A

[3] We are identifying x and $\{x\}$ here.

[4] for $\nu'(l)$ where l satisfies $l(-1) = x'$, $l(0) = y'$ and $l(1) = z'$

cell in state (E, y) (or acting as such) treats a neighbor in state $x \in A \cup \{\Box\}$ as if that were in state (E, x). For all other local configurations ν' can be defined arbitrarily because they will never occur in configurations belonging to simulations of M.

In an existential step cells storing a universal state from Q only collect the states of their neighbors and alter their flag, resulting in a state (U, xyz). Cells in a state (U, y) "did their work" during the previous existential step and do nothing except changing their flag during a universal step. In other cases the cells of M' work as they would in C and alter their flag.

Note that for an arbitrary configuration of M' which occurs during a computation starting with some initial configuration c_w either all flags are E or all flags are U, hence either all non-quiescent states are existential or they are all universal.

Now suppose M makes one $\exists\forall$-step $c^N \vDash^{\exists}_{\exists(c)} \bar{c} \vDash^{\forall}_{\forall(c)} \hat{C}$. Denote by $\mathrm{E} \otimes C$ the set of configurations of M' which is obtained from the set C of configurations of M by adding the E flag in every non-quiescent cell in every configuration, and analogously for $U \otimes C$. Because of the definition of ν' and since $\exists(\mathrm{E} \otimes c) = \mathbf{Z}$ for M' one gets $(\mathrm{E} \otimes c)^N \vDash^{\exists}_{\exists(\mathrm{E}\otimes c)} \mathrm{U} \otimes \bar{c}$, and since $\forall(\mathrm{E} \otimes c) = \emptyset$ furthermore $\mathrm{U} \otimes \bar{c} \vDash^{\forall}_{\forall(\mathrm{E}\otimes c)} \mathrm{U} \otimes \bar{c}$. I.e. for M' holds: $\mathrm{E} \otimes c \vDash^{\exists\forall} \mathrm{U} \otimes \bar{c}$.

An analogous consideration (which is a bit more complicated because $\Box$ is existential) shows, that for M' also holds $\mathrm{U} \otimes \bar{c} \vDash^{\exists\forall} \mathrm{E} \otimes \hat{C}$. That is, every M-step $c \vDash^{\exists\forall} \hat{C}$ can be simulated by 2 M'-steps

$$\mathrm{E} \otimes c \vDash^{\exists\forall} \mathrm{U} \otimes \bar{c} \vDash^{\exists\forall} \mathrm{E} \otimes \hat{C} \ .$$

Conversely, for 2 steps of M' starting in a configuration of the form $\mathrm{E} \otimes c$, there has to be a $\bar{c}$ such that

$$\mathrm{E} \otimes c \vDash^{\exists\forall} \mathrm{U} \otimes \bar{c} \vDash^{\exists\forall} \mathrm{E} \otimes \hat{C} \ .$$

It is then not difficult to see that this implies that for M one has

$$c^N \vDash^{\exists}_{\exists(c)} \bar{c} \vDash^{\forall}_{\forall(c)} \hat{C} \quad \text{i.e.} \quad c \vDash^{\exists\forall} \hat{C} \ .$$

Hence, given an input word, for every accepting computation tree for M of height h there is one for M' of height $2h$ and vice versa. ∎

Essentially the same argument as above leads to the following result:

Theorem 3.2 *For all time bounds t and space bounds s holds:*

$$\exists\forall\text{–ACA–TIME–SPACE}(\mathrm{O}(t), s) = \forall\exists\text{–ACA–TIME–SPACE}(\mathrm{O}(t), s) \ .$$

4 Uniform ACA versus bounded fan-in circuits

Uniform ACA have been defined in [3] as follows. A deterministic transition function $\delta : Q^N \to Q$ is said to be *compatible* with a nondeterministic transition function $\nu : Q^N \to 2^Q$, written as $\delta \in \nu$ for short, iff for all $l \in Q^N$: $\delta(l) \in \nu(l)$. A configuration c of a UACA is called *existential* (resp. *universal*) iff c_1 is existential (resp. universal). A tree of configurations is a computation tree for a UACA iff each existential configuration c has exactly one successor c' such that $c \vdash_\delta c'$ for some $\delta \in \nu$ and for each universal configuration c the set of successors is $\mathrm{Succ}(c) = \{c' \mid$

there is a $\delta \in \nu : c \vdash_\delta c'\}$. Hence the main difference to nonuniform ACA is that if in a configuration c the same local configuration occurs several times, in a UACA all cells observing it will enter the *same* new state, while in a nonuniform ACA they don't have to.

For completeness we mention the so-called weak ACA (WACA). They are "essentially" CA with deterministic cells with the exception of one cell, say cell 1, which is an alternating one. WACA are time equivalent to UACA [3].

One of the main contributions of this paper are results concerning the relations between alternating CA and uniform circuit families (UCIR). In this section UACA will be shown to be time equivalent to UCIR with bounded fan-in gates. As a corollary one also gets the time equivalence of UACA and ATM, solving a problem left open in [3]. In order to have the same set of input symbols for all models, we restrict ourselves to the input alphabet $A = \{0, 1\}$ below.

A circuit family with bounded fan-in consists of a circuit C_n with n inputs and one output for each integer $n \geq 1$, such that each C_n consists of $\neg$-gates (having one entry[5]) and $\wedge$- and $\vee$-gates with two entries. It is sometimes convenient to assume that there are "gates" producing the constants 0 (resp. 1) as output. Such devices obviously can be built from the $\neg$-, $\wedge$- and $\vee$-gates using one input bit. Furthermore we assume that the n input bits are provided at the exits of "input gates" (having no "entries") and that the result of the circuit is available at the exit of an "output gate" (having one "entry" and doing nothing).

The size of a circuit is the total number of entries of all gates. For circuits with bounded fan-in this differs from the number of gates only by a constant factor (which we will ignore throughout the paper) and has the advantage that it actually is the definition used for circuits with unbounded fan-in gates (see next section). The depth of a circuit is the length of the longest path from an input to the output. These notions are generalized to circuit families in the obvious way.

An important topic in the definition of UCIR is a concept also called *uniformity* (which has nothing to do with the uniformity condition for ACA). Different versions are used in the literature; see [5] for an overview. In this paper we will use the following (is also applicable to UCIR with unbounded fan-in): To the gates of each circuit C_n numbers v of length $O(\log \text{size}(C_n))$ have to be assigned such there is a deterministic TM which on input of (n, v, i) can compute the type of gate v, its number e of entries and for $i \leq e$ the number v' of the gate providing the input for the i-th entry of v, using space complexity $O(\log \text{size}(C_n))$.

Lemma 4.1 *For functions $t \geq n$ and $s \geq n$ which can be computed in space $O(s)$:*

$$\text{UACA-TIME-SPACE}(O(t), O(s)) \subseteq \text{UCIR-DEPTH-SIZE}(O(t), 2^{O(s)})$$

Proof. Let C be an arbitrary UACA. For each input size n a circuit C_n will be constructed which accepts an input word w of length n, if and only if C accepts w, such that the resulting circuit family is uniform.

The circuit consists of an "upper" and a "lower" part. The upper part of the circuit is similar to the construction for the proof of [5, Theorem 3] and checks for all $w \in A^n$ whether C accepts them in time $t(n)$ or not: For all $1 \leq i \leq t(n)$

[5]Sometimes we don't use the word input to avoid confusions with the inputs of the circuit, and analogously for "exit" instead of output of a gate.

and all configurations c of C occupying space $s(n)$ or less – there are $2^{O(s(n))}$ such configurations – the circuit contains a gate labeled (i,c). Imagine the gates being arranged in $t(n)$ levels with an increasing first index from the bottom to the top. Furthermore there is a zero level with 2^n gates labeled $(0,c_w)$ where the c_w are the initial configurations of C for inputs $w \in A^n$. For all gates their type is $\wedge$ (resp. $\vee$) if the configuration is universal (resp. existential). The inputs of a gate (i,c) are the outputs of all gates $(i+1,c')$ such that c' is a successor configuration of c. In general there are more than two successors of a configuration c, but for the UACA there are not more than $K := |Q|^{(|Q|^3)}$, i.e. a constant number. So the 'gates' used above are implemented as binary trees of height $\leq \lceil ld(K) \rceil$ consisting of ordinary gates with in-degree 2. Thus the overall depth of the circuit is increased only by a constant factor that depends on the machine C and not on the length of the input. As the circuit is constructed from lower levels to higher levels a 'gate' labeled with an accepting configuration is replaced by a constant 1 and 'gates' labeled with a configuration using more space than $s(n)$ and gates on level $t(n)+1$ are replaced by a constant 0. Now the following can be shown by induction [5]: the output of a 'gate' $(0,c_w)$ is 1 iff there is an accepting tree for C on w of height $\leq t(n)$.

The lower part of C_n uses the actual input bits $x_1,\ldots,x_n$ to select the proper 'gate' $(0,c_w)$ the output of which is used as the output bit of the whole circuit: To this end there are 2^n "comparators" K_b each comparing $w = x_1 \cdots x_n$ to one of the constant bit pattern $b = b_1 \cdots b_n \in A^n$ producing a 1 as output iff $w = b$. The outputs of K_b and the gate labeled $(0,c_b)$ are fed to an $\wedge$-gate. The outputs of these 2^n $\wedge$-gates are fed to a tree of $\vee$-gates (with height $ld(2^n) = n$). The root of this tree is the output of the circuit and the following holds: The output of the constructed circuit is 1 iff C accepts w in time $\leq t(n)$.

The depth of the circuit is $\lceil ld(K) \rceil \cdot (t(n)+2) + \lceil ld(n+1) \rceil + 1 + n \in O(t(n))$. The size of the circuit is dominated by the size of its upper part which is $2^{O(s(n))} \cdot (t(n)+1)$. Since $t(n) \leq 2^{O(S(n))}$, the overall size is $2^{O(s(n))}$. ∎

Note that the assumption $t(n) \geq n$ is no real restriction because a UACA needs at least n steps until every input symbol might have had an influence on the state of the origin cell.

Since it is known [5], that for t and s which can be linearly approximated[6]:

$$\text{UCIR–DEPTH–SIZE}(O(t), 2^{O(s)}) \subseteq \text{ATM–TIME–SPACE}(O(t), O(s))$$

and that [3]

$$\text{ATM–TIME–SPACE}(O(t), O(s)) \subseteq \text{UACA–TIME–SPACE}(O(t), O(s)) \ ,$$

one also gets the opposite inclusion as in Lemma 4.1 and therefore:

Theorem 4.2 *For $t(n) \geq n$ which can be linearly approximated holds:*

$$\text{UACA–TIME}(O(t)) = \text{ATM–TIME}(O(t)) = \text{UCIR–DEPTH}(O(t)) \ .$$

[6] I.e. there is a $\hat{t}$, $t \leq \hat{t} \in O(t)$, (resp. $\hat{s}$) which is time constructible.

5 Nonuniform ACA and circuits

In the previous section uniform ACA have been shown to be time equivalent to circuit families with bounded fan-in. In the following an analogous relation between nonuniform ACA and circuit families with unbounded fan-in will be established. ACA will also be compared to circuits with bounded fan-in.

In a circuit with unbounded fan-in (UBUCIR) $\wedge$- and $\vee$-gates are allowed with an arbitrary number of entries. The uniformity condition used is exactly the same as for circuit with bounded fan-in above.

The proof in section 4 cannot be applied unchanged to nonuniform ACA because unlike for UACA the number of successors of a configuration of a nonuniform ACA ($\exists\forall$–ACA or $\forall\exists$–ACA) cannot be bounded by a constant[7]. This leads to the idea to compare nonuniform ACA with uniform circuit families with unbounded fan-in gates.

Lemma 5.1 *For functions $t \geq n$ and $s \geq n$ which can be computed in space $\mathrm{O}(s)$:*

$$\exists\forall\text{–ACA–TIME–SPACE}(\mathrm{O}(t), \mathrm{O}(s)) \subseteq \text{UBUCIR–DEPTH–SIZE}(\mathrm{O}(t), 2^{\mathrm{O}(s)})$$

Proof. Recall the proof of Lemma 4.1. Here, a similar construction is used, but there are the following changes: A gate labeled (i, c) in the upper part of the circuit now is replaced by a tree of height 2 of gates with unbounded fan-in. For $\exists\forall$–ACA the root of the tree is a $\vee$ gate whose inputs are connected to intermediate $\wedge$ nodes which correspond to the $\bar{c}$ such that $c^N \vDash^{\exists}_{\exists(c)} \bar{c}$. The inputs of these intermediate gates in turn are connected to the outputs of the gates $(i+1, \hat{c})$, for all $\hat{c} \in \hat{C}$, where $\bar{c} \vDash^{\forall}_{\forall(c)} \hat{C}$. Again, gates labeled with accepting configurations are replaced by a constant 1 and gates labeled by a configuration using more than space $s(n)$ or gates on level $t(n)+1$ are replaced by 0. It can be shown by induction on the level i where a gate (i, c) occurs that it has output 1 if and only if c is the root of an accepting computation tree of height $\leq t(n) - i$ of the $\exists\forall$–ACA. Therefore the output of a gate labeled $(0, c_w)$ of the $\exists\forall$–ACA is 1 iff it accepts c_w.

The comparators selecting the output of the proper gate $(0, c_w)$ can be implemented using a single $\wedge$-gate with fan-in $2n$, making use of input symbols and their negations. Likewise the outputs of these $\wedge$-gates are input to a single $\vee$-gate with fan-in 2^n. Thus the depth of the circuit with n inputs is $2t(n)+2$. ∎

If in the above construction gates with large fan-in are replaced by trees of gates with fan-in 2 one gets the following result.[8]

Lemma 5.2 *For functions t and s satisfying the requirements of Lemma 5.1 holds:*

$$\begin{aligned} \exists\forall\text{–ACA–TIME–SPACE}(\mathrm{O}(t), \mathrm{O}(s)) &\subseteq \text{UCIR–DEPTH–SIZE}(\mathrm{O}(st), 2^{\mathrm{O}(s)}) \\ \exists\forall\text{–ACA–TIME}(\mathrm{O}(t)) &\subseteq \text{UCIR–DEPTH}(\mathrm{O}(t^2)) \end{aligned}$$

Proof. First, the same construction as above is used. Then, each gate with a fan-in $f > 2$ is replaced by a tree of height $\lceil ld(f) \rceil$ of gates with fan-in 2. The maximum

[7] It depends on the number of cells in a configuration that are in a universal state observing a local configuration which allows at least two new states, i.e. $|\mathrm{Succ}(c)|$ depends on the space complexity and in general cannot be bounded by a constant.

[8] In the light of [5, Theorem 3] this follows from [3, Theorem 1], but the proof of the latter needs some modifications in our opinion (see [4]).

fan-in cannot exceed the number of gates in the UBUCIR-circuit[9] and the constructed circuits contain $2^{O(s(n))} \cdot (t(n)+1)$ gates, where $2^{O(s(n))}$ is the number of possible configurations occupying space $O(s(n))$). Since $t \leq 2^{O(s(n))}$ (see [1] for a similar result for ATM), the number of gates and thus the maximum fan-in is bounded by $2^{O(s(n))}$. Therefore the depth of the UCIR-circuit is bounded by $s(n)t(n)$ which is $O(t^2(n))$. ∎

We are now going to prove a kind of reverse result compared to Lemma 5.1, thus establishing a close relation between nonuniform ACA and circuit families with unbounded fan-in (satisfying certain conditions).

Lemma 5.3 *Let $t \geq n$ and $s \geq n$ be functions which can be computed in space $O(s)$ and which satisfy $t \in \Omega(s)$. Then it holds:*

$$\text{UBUCIR-DEPTH-SIZE}(O(t), 2^{O(s)}) \subseteq \exists\forall\text{-ACA-TIME-SPACE}(O(t), O(s))$$

Proof. Let C_n be the circuit with n inputs, having depth $O(t)$ and size $2^{O(s)}$. We have to construct an $\exists\forall$-ACA accepting the same $w \in A^n$ as C_n, satisfying the time and space bounds $O(t)$ and $2^{O(s)}$. The construction is modeled on the one in the proof for the analogous simulation of UCIR by ATM [5, Theorem 4]. It essentially relies on a procedure **value** with two integer parameters v and i. Its task is to determine whether the i-th input of gate v will get a 1 as input. Consider the following algorithm sequentially executing two steps:

A) universally do in parallel:

- **1)** $v' \leftarrow$ **guess** existentially the number of a gate v'
- **2)** $\tau' \leftarrow$ guess existentially the type of v'

B) universally do in parallel:

- **1)** check that during **guess** in **A1)** above all participating cells guessed that they should do something existentially
- **2)** do sequentially:
  - **21)** check that the output of v' is the i-th input of v
  - **22)** check that v' is of type τ'
- **3)** depending on the type τ' do one of the following:
  - **31)** for inputs:
  - **32)** if $\tau' = \vee$: sequentially do
    - **a)** $i' \leftarrow$ **guess** existentially a number
    - **b)** universally do in parallel
      - **i)** compute **value**(v', i')
      - **ii)** check that during **guess** in **B32a)** above all participating cells guessed that they should do something existentially
  - **33)** if $\tau' = \wedge$: analogously to the case $\tau' = \vee$.

The crucial point of the construction is to make sure that only a constant number of steps is needed between the start of **value** and the recursive calls happening inside.

[9] Note, that its size is $\geq$ its depth which in turn is $\geq$ the time complexity of the ACA which is $\geq n$.

For this to be true, some numbers $(v',\ i')$ have to be guessed in constant time, although in general they will consist of $k = \log \mathrm{size}(C_n)$ many bits.

This is achieved by using k subsequent cells, each of which guesses one bit. But there is a further complication. Sometimes a number has to be guessed existentially, e.g. in **B32a)**, but sometimes numbers have to be guessed universally, e.g. in the not shown step **B33a)**. The information which is the case is present at some cell, but it cannot be provided to all k guessing cells in constant time. The solution is as follows: Guessing is done in two steps. In the first one, each of the k cells guessing whether the number guessing has to be an existential one or a universal one and enter an existential or universal state accordingly. In the second step each cell guesses a bit. Afterwards, see **B32bii)** above, it is verified that all cells have correctly guessed that an existential guess was needed. This can be done in k steps by sending τ' to the guessing cells. For the universal guessing of numbers an analogous approach can be used.

This way the next recursion level is always called after a constant number of steps, and the time for verification only contributes a *summand* of $\log \mathrm{size}(C_n) \in \mathrm{O}(s)$ to the overall execution time. All other considerations are analogous to the proof in [5]. ∎

As a corollary one obtains:

Theorem 5.4 *For functions t and s satisfying the requirements of Lemmata 5.1 and 5.3 holds:*

$$\exists\forall\text{-}\textsc{Aca}\text{-}\textsc{Time}\text{-}\textsc{Space}(\mathrm{O}(t), \mathrm{O}(s)) = \textsc{UbUcir}\text{-}\textsc{Depth}\text{-}\textsc{Size}(\mathrm{O}(t), 2^{\mathrm{O}(s)}) \ .$$

It is not known whether UbUcir of depth $\mathrm{O}(d)$ can be simulated by Ucir of depth $\mathrm{O}(d)$. The general expectation seems to be that this is not possible. Therefore Lemma 5.3 together with Theorem 4.2 can be taken as an indication, that a linear-time simulation of nonuniform Aca by Atm is "unlikely" to exist.

Furthermore one should note that in the comparisons of UbUcir and Atm (instead of Aca) the depth of the circuits usually corresponds to number of alternations and *not* to the time of the Atm. To put it the other way round, alternation depth of Atm is linearly related to the time of Aca.

6 Conclusions

The results obtained above together with those proved in [3] lead to the relatively simple situation depicted in Figure 1. An arrow from A to B with a label $\mathrm{O}(f(t))$ indicates that each machine of type A with time complexity t can be simulated by a machine of type B with time complexity $\mathrm{O}(f(t))$. Thick lines indicate results obtained in this paper.

Thus, as long as the time bound satisfies some standard requirements, uniform Aca correspond to circuit families with bounded fan-in and nonuniform Aca correspond to circuit families with unbounded fan-in.

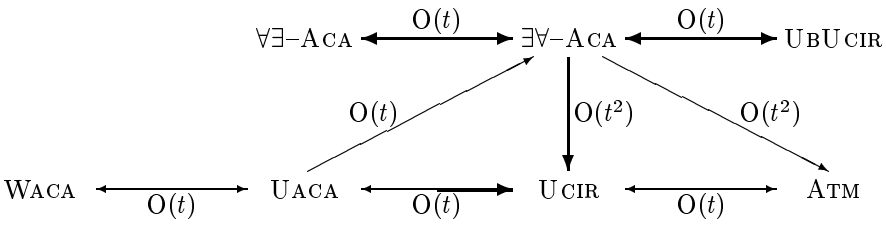


Figure 1: Simulations between different ACA, UCIR and ATM

References

[1] Ashok K. Chandra, Dexter C. Kozen, and Larry J. Stockmeyer. Alternation. *Journal of the ACM*, 28(1):114–133, 1981.

[2] Kamala Krithivasan and Meena Mahajan. Nondeterministic, probabilistic and alternating computations on cellular array models. *Theoretical Computer Science*, 143(1):23–49, May 1995.

[3] Martín Matamala. Alternation on cellular automata. *Theoretical Computer Science*, 180(1-2):229–241, 1997.

[4] Frank Reischle and Thomas Worsch. Simulations between alternating CA, alternating TM and circuit families. Technical Report 9/98, Fakultät für Informatik, Universität Karlsruhe, 1998.

[5] Walter L. Ruzzo. On uniform circuit complexity. *Journal of Computer and System Sciences*, 22(3):365–383, 1981.

Computation Theory of Cellular Automata

Klaus Sutner
Carnegie Mellon University
Pittsburgh, USA
http://www.cs.cmu.edu/~sutner

August 14, 1998

Abstract

Cellular automata provide a specialized model of parallel computation and give rise to a number of typically difficult classification problems. We discuss the computational aspects of the reversibility problem for a class of very simple additive cellular automata. On the other end of the spectrum, we study the computatonial properties of Gandy Machines, a very general model of parallel computation.

1 Introduction

One of main areas of interest in the study of cellular automata is a comparison to other models of computation. From the foundational work of Church, Gödel, Turing and others it is apparent that Turing machines provide a natural and adequate basic model of computation. By imposing time and space bounds, and perhaps with the addition of nondeterminism, one obtains an elaborate system of complexity classes that serve as a classification scheme for computational problems, at least with respect to sequential computation.

As was recognized during the last two decades, even exceedingly simple cellular automata turn out to be difficult to analyze computationally. As a case in point, we will discuss a family of additive cellular automata over the two-element field $\mathbb{F}_2$ on two-dimensional grids. These automata are far removed from computational universality, and are very much predictable in the sense that elements in the orbit of a configuration can be easily computed without direct simulation. As Fredkin noted, these automata give a somewhat unintentional solution to von Neumann's problem of constructing a self-reproducing machine. A lot of information about the structure of the phase spaces of these automata can be determined using various algebraic methods, see [9]. However, finding efficient methods to determine the reversibility of these automata is difficult. One encounters problems of an algebraic nature: one has to determine the location of irreducible polynomials over $\mathbb{F}_2$ in a sequence of polynomials, a problem that currently seems to yield only to a brute force computation.

At the other extreme, we will discuss a very general model of synchronous parallel computation due to Gandy [4, 17, 18]. Gandy's approach naturally encompasses other parallel models such as cellular automata, families of circuits and various parallelized versions of RAMs. Gandy's mechanisms are premised on the idea of parallel devices that use only local information about their inputs, and process this information essentially by a table lookup, very much in the spirit of cellular automata. We will analyze the complexity of calculating in a Gandy Machine and comment on the meaning of locality in this context.

Undefined notions of recursion theory or complexity theory can be found in [12, 11, 7] or the references quoted there.

2 Predictability and Reversibility in Xor Automata

Consider a finite cellular automaton $\langle \mathcal{C}, \rho \rangle$ over some grid $G = \langle V, E \rangle$, an undirected graph. Thus, the configurations are maps $X : V \to \Sigma$ into the alphabet of the automaton, and the local maps a defined on local configurations obtained by restriction to the neighborhood of a point in the graph. It is clear that the problem of calculating $\rho^t(X)$, often referred to as the Prediction Problem, is in general P-complete, even if we restrict the graph to be a simple path. It is straightforward to express a cellular automaton as a circuit of polynomial size, but there is no hope to obtain exponential speed-up in general, unless, of course, some collapse of complexity classes such as P = NC occurs.

The question of the existence of predecessors naturally leads to nondeterministic complexities. The finite case often reflects the difficulty of the associated infinite problem. In the case of Predecessor Existence, the problem is undecidable for infinite 2-dimensional automata, even if the target configuration is finite, see [25]. For infinite 1-dimensional automata, Predecessor Existence can be expressed as a path existence problem in the de Bruijn graph (or rather, its square) of the cellular automaton, and is easily decidable in quadratic time [21]. Note, though, that the infinitary version of the problem admits some variations that appear to have no clear counterpart in the finite case. For example, we can restrict the target configuration to be recursive.

Theorem 2.1 *For an infinite 1-dimensional cellular automaton, every recursive target configuration that has a predecessor already has a recursive predecessor. However, the existence of such a predecessor is undecidable, and if a predecessor exists, it cannot be computed effectively in general.*

At any rate, a similar discrepancy betweem dimensions 1 and 2 occurs for finite cellular automata. In the 2-dimensional case, the problem is NP-complete, even for a fixed rule (the obvious transformation from a tiling problem requires a family of rules, see [22] for a reduction directly from 3-SAT). For 1-dimensional automata the complexity drops to NL, and the problem is hard for this class with respect to log-space reductions.

Determining membership of a target configuration in the orbit of a given source configuration on a finite cellular automaton is PSPACE-complete: since cellular automata can simulate Turing machines, Linear Bounded Automata Acceptance is a special case of this problem. If we consider the structure of orbits in general, the threshold to undecidability is readily crossed, even if the graphs are restricted to simple paths or circles. Since the configuration space of a single such automaton is finite, it is necessary to consider a family $\langle \mathcal{C}_n, \rho \rangle$, where, say, G_n is a path or circle of length n, and the same local rule is uniform for all n. This is quite similar to the notion of uniform families of circuits in parallel complexity. In analogy to Wolfram's classes, let us say that the family is in class $\mathrm{BO}(f(n))$ if all configurations on $\langle \mathcal{C}_n, \rho \rangle$ evolve to a configuration with period of length $O(f(n))$. Thus, the length of limit cycles is $O(f(n))$. We write BO^0 for the special case where all configurations evolve to a fixed point. Clearly, every such cellular automaton belongs to $\mathrm{BO}(2^{O(1)})$, but for smaller growth rates it becomes difficult to determine membership. More precisely, the following results are shown in [22]:

Theorem 2.2 *Class* BO^0 *is* Π_1^0*-complete. The classes* $\mathrm{BO}(n^k)$ *are* Σ_2^0*-complete, even for* $k = 0$.

Using Σ_2^0-uniformization, it follows from the second result that there is a Σ_2^0 function that bounds the lengths of limit cycles, but no such bounding function will be recursive in general. It is worth mentioning that these results do not directly follow from the usual observations about simulations of Turing machines by cellular automata. The computations of a Turing machine deal only with instantaneous descriptions of a certain restricted form. In our situation, however, we have to contend with arbitrary configurations, most of which are meaningless from the point of view of the simulated Turing machine. It is easy to weed out configurations that are syntactically of the wrong form, but it is difficult to deal with configurations that correspond to syntactically correct instantaneous descriptions which happen not to occur in any computation of the corresponding Turing machine (this property is undecidable). The device used in the reference is a self-verifying Turing machine, a machine that performs many redundant computations and that will recognize and discard inappropriate configurations after finitely many steps. Similar techniques where also used by Shepherdson [15]. It is shown there, for example, that the question of whether a configuration appears in the orbit of another can be made to be of any recursively enumerable degree, by proper choice of the corresponding Turing machine. For a different approach to the problem of dealing with erroneous configurations see [1].

2.1 Xor Automata

When the local map is constrained to be a simple algebraic operation, the situation chances drastically. We will only consider the arguably simplest situation here, with alphabet $\Sigma = \{0, 1\}$ and the local map given by addition modulo 2.

We denote this rule by σ and refer to these cellular automata as σ-automata. A first extensive study of the phase space in particular of 1-dimensional σ-automata can be found in [9].

First off, prediction of $\langle \mathcal{C}, \sigma \rangle$ can be handled by matrix multiplication over $\mathbb{F}_2$, the two-element field. More precisely,

$$\sigma^t(X) = M^t \cdot X$$

where M is the adjacency matrix of the underlying graph. Matrix multiplication is well known to be in parallel logarithmic time, and M^t can be computed in $O(\log t)$ multiplication steps by the usual divide-and-conquer approach. Hence we have

Observation: For a σ-automaton, prediction is in NC^2.

As a matter of fact, a little calculation with binomial coefficient shows that prediction is even in NC^1 for paths and circles. These results were generalized to Abelian groups and other algebraic structures in [10].

How about the existence of predecessors? This is again a problem of linear algebra, but when we search for predecessor of small cardinality, the problem becomes NP-hard for general graphs, see [20].

Likewise, the question of reversibility of a σ-automaton is equivalent to the matrix M being non-singular, the latter property being in NC. We will now focus on a more restricted class of automata, where the underlying graph is standard two-dimensional m by n grid. We denote these graphs by $P_{m,n}$. Furthermore, we use standard von Neumann neighborhoods. More precisely, the automata which include the center cell from the summation will be referred to as σ^+-automata, whereas σ-automaton denotes a machine on $P_{m,n}$ where the center cell is excluded. We assume fixed boundary conditions, though the results mentioned below all have counterparts for circular boundary conditions.

Since the global function $\sigma : \mathcal{C} \to \mathcal{C}$ is a linear map, we may consider $d(m,n)$, the corank of σ. One can easily see that $d(m,n) \leq \min(m,n)$.

Theorem 2.3 *A σ-automaton on a $m \times n$ grid is reversible iff $m+1$ and $n+1$ are coprime. More precisely, $d(m,n) = \gcd(m+1, n+1) - 1$.*

Hence, we can test reversibility in time polynomial in $\log n$, the size of the succinct version of problem.

2.2 σ^+-Automata

Surprisingly, for σ^+-automata no such simple procedure seems to exist. To get some understanding of the complexity of determining the reversibility of a σ^+-automaton, we define a binary version of the Chebyshev polynomials of the second kind. More precisely, let $\pi_0 = 0$ and π_i is a polynomial defined by

$$\pi_i(x) = U_{i-1}(x/2) \bmod 2 \in \mathbb{F}_2[x].$$

Alternatively, these polynomials can be defined in terms of a homogeneous second order recurrence over $\mathbb{F}_2[x]$:

$$\pi_{n+2} = x \cdot \pi_{n+1} + \pi_n$$

with initial conditions $\pi_0 = 0$, $\pi_1 = 1$. Note that second order recurrences similar to the last one can also be used to construct reversible cellular automata, a trick going back to Fredkin, see for example [23].

As one might suspect from the simplicity of the recurrence, the π-polynomials enjoy many special divisibility properties.

Proposition 2.1 *Let $a, b \in F[x]$ be coprime, where F is an arbitrary finite field. Define a sequence (τ_n) in $F[x]$ by $\tau_0 = 0$, $\tau_1 = 1$, and $\tau_n = a \cdot \tau_{n-1} + b \cdot \tau_{n-2}$, for all $n \geq 2$. Then*

$$\begin{aligned} \tau_p &= \tau_{q+1}\tau_{p-q} + b \cdot \tau_q \tau_{p-q-1}, \\ \gcd(\tau_n, \tau_m) &= \tau_{\gcd(n,m)}, \\ \tau_{2^k n} &= a^{2^k-1}\, \tau_n^{2^k}, \\ \tau_{2n+1} &= \tau_{n+1}^2 + b \cdot \tau_n^2. \end{aligned}$$

where $p \geq q + 1$.

Our binary Chebyshev polynomials correspond to $a = x$ and $b = 1$, so that the last two equations can also be written as

$$\begin{aligned} \pi_{2^k n} &= x^{2^k-1}\, \pi_n^{2^k}, \\ \pi_{2n+1} &= \pi_{n+1}^2 + \pi_n^2 \\ &= (\pi_{n+1} + \pi_n)^2. \end{aligned}$$

As a consequence of the second equation, we have $m \mid n \iff \pi_m \mid \pi_n$.

A recent paper by Barua and Ramakrishnan [2, 14] shows that m by n grids under rule σ^+ are reversible iff the two polynomials $\pi_{m+1}(x)$ and $\pi_{n+1}(x+1)$ are coprime. One can extend their results to show that π_n is the minimal polynomial of the map $\sigma^+ : \mathcal{C}_n \to \mathcal{C}_n$ (underlying graph a path of length n). From this one obtains

Lemma 2.1

$$d^+(m,n) = \deg \gcd\big(\pi_{m+1}(x), \pi_{n+1}(x+1)\big)$$

where $d^+(m,n)$ is corank of the global map $\sigma^+ : \mathcal{C} \to \mathcal{C}$.

The analogous result for plain σ-automata is $d(m,n) = \deg \gcd\big(\pi_{m+1}, \pi_{n+1}\big)$. How does this help to determine reversibility? Needless to say, one does not have run the recurrence n steps to obtain π_n. In fact,

$$\pi_n(x) = \sum_i \binom{n+i}{2i+1} x^i \bmod 2.$$

By Lucas' theorem [8], $\binom{x}{y} \bmod 2 = \prod \binom{x_i}{y_i} \bmod 2$, where x_i is the i-th digit in the binary expansion of x, and likewise for y, so the coefficients of π_n can be computed in time $O(\log n)$. But the polynomials are still objects of size linear in n.

To get more information about the possible values of the GCD of two such polynomials we have to consider their irreducible factors. Define the *depth* of an irreducible polynomial $\tau \in \mathbb{F}_2[x]$ to be

$$\mathrm{dep}(\tau) = \min\bigl(n > 0 \bigm| \tau \text{ divides } \pi_n \bigr).$$

It is not hard to see that every irreducible polynomial has a depth. Now let

$$\rho_n = \prod_{\mathrm{dep}(\tau)=n} \tau^2$$

be the squared product of all irreducible factors that occur at level n for the first time. For the sake of completeness, let $\rho_1 = \pi_1 = 1$. We will refer to ρ_n as the *critical factors* of π_n. The following decomposition theorem describes the structure of a π-polynomial in terms of critical factors, see [19].

Theorem 2.4 *For all positive $n = 2^k p$, where p is odd,*

$$\pi_n(x) = x^{2^k-1} \prod_{d|p} \rho_d^{2^k}(x) = x^{2^k-1} \prod_{d|p} \rho_d(x^{2^k}).$$

Furthermore, $\deg \rho_d = \phi(d)$ *unless* $d = 1$.

Here ϕ is Euler's totient function. Möbius inversion allows one to compute the critical factors in terms of the π-polynomials: $\rho_n = \prod_{d|n} \pi_{n/d}^{\mu(d)}$, again for all odd numbers n, where μ denotes the Möbius function. As an example of a decomposition consider π_{300}. We have the factorization

$$\begin{aligned}\pi_{300} = {} & x^3 (1+x)^8 (1+x+x^2)^8 (1+x^3+x^4)^8 \\ & (1+x+x^2+x^3+x^5+x^6+x^{10})^8 \\ & (1+x^3+x^4+x^5+x^9+x^{12}+x^{13}+x^{15}+x^{20})^8\end{aligned}$$

where the irreducible terms are associated with divisors $2, 3, 5, 15, 25$, and 75, respectively. All critical factors are squares of just one irreducible term in this case, but, e.g., $\rho_{17} = \left(1+x+x^4\right)^2 \left(1+x+x^2+x^3+x^4\right)^2$.

To use the lemma, we have to determine the depth of polynomials of the form $\tau(x+1)$. More precisely, given irreducible τ of depth d, we would like to calculate the depth of the image of τ under the involution $x \mapsto x+1$. We can pin down the depth of a polynomial to some degree, but not much else is known. Recall that the suborder of 2 in the multiplicative group $\mathbb{Z}_n^*$, n odd, is defined as $\mathrm{sord}_n(2) = \min\bigl(i \bigm| 2^i \equiv \pm 1 \pmod n \bigr)$.

Theorem 2.5 *Let $\tau \in \mathbb{F}_2[x]$ be an irreducible polynomial of degree k and depth d. Letting $q = 2^k$, d divides $q-1$ iff the linear term in τ vanishes, and $q+1$ otherwise. In either case, k is the suborder of 2 in the multiplicative group $\mathbb{Z}_d^*$.*

Still, for special values of m an n these results suffice to solve the problem of determining reversibility of a σ^+-automaton.

Corollary 2.1 *For $m+1 = 2^k$ and $n+1 = 2^l p$, p odd, we have*

$$d^+(n,m) = \begin{cases} 0 & \text{if } 3 \not| n+1, \\ 2^{l+1} & \text{if } l < k-1, \\ 2^k - 1 & \text{otherwise.} \end{cases}$$

Recently, Sarkar [13] has shown that no square grids are totally irreversible (meaning $d^+(n,n) = n$), other than for $n = 4$.

However, in general there appears to be no easy way to compute the depth of an irreducible polynomial. More precisely, the depth function appears to display fairly chaotic behavior under the involution $x \mapsto x+1$.

Problem: Find an efficient way to compute the depth of $\tau(x+1)$, for an irreducible polynomial τ of depth n.

Problem: Generalize these results to more general algebraic structures, such as Abelian groups, groups, semigroups, and so forth. See [10] for some results in the 1-dimensional case.

The following table exhibits some of the chaotic behavior of the depth function in conjunction with the involution on $\mathbb{F}_2[x]$. It shows the depths of all $\tau(x+1)$ where τ is irreducible of degree 8.

number	image	depth	irreducible polynomial
1	17	257	$1+x+x^3+x^4+x^8$
2	12	51	$1+x^2+x^3+x^4+x^8$
3	15	257	$1+x+x^3+x^5+x^8$
4	29	255	$1+x^2+x^3+x^5+x^8$
5	13	255	$1+x^3+x^4+x^5+x^8$
6	10	257	$1+x+x^2+x^3+x^4+x^5+x^8$
7	16	255	$1+x^2+x^3+x^6+x^8$
8	23	257	$1+x+x^2+x^3+x^4+x^6+x^8$
9	9	257	$1+x+x^5+x^6+x^8$
10	6	255	$1+x^2+x^5+x^6+x^8$
11	28	255	$1+x^3+x^5+x^6+x^8$
12	2	85	$1+x^4+x^5+x^6+x^8$
13	5	257	$1+x+x^2+x^4+x^5+x^6+x^8$
14	24	257	$1+x+x^3+x^4+x^5+x^6+x^8$
15	3	257	$1+x+x^2+x^7+x^8$
16	7	257	$1+x+x^3+x^7+x^8$
17	1	51	$1+x^2+x^3+x^7+x^8$
18	19	257	$1+x+x^2+x^3+x^4+x^7+x^8$
19	18	257	$1+x+x^5+x^7+x^8$
20	21	85	$1+x^3+x^5+x^7+x^8$
21	20	255	$1+x^4+x^5+x^7+x^8$
22	22	255	$1+x^2+x^3+x^4+x^5+x^7+x^8$
23	8	257	$1+x+x^6+x^7+x^8$
24	14	257	$1+x+x^2+x^3+x^6+x^7+x^8$
25	30	257	$1+x+x^2+x^4+x^6+x^7+x^8$
26	27	85	$1+x^2+x^3+x^4+x^6+x^7+x^8$
27	26	257	$1+x+x^2+x^5+x^6+x^7+x^8$
28	11	257	$1+x+x^4+x^5+x^6+x^7+x^8$
29	4	85	$1+x^2+x^4+x^5+x^6+x^7+x^8$
30	25	255	$1+x^3+x^4+x^5+x^6+x^7+x^8$

Table 1: The depths of all irreducible binary polynomials τ of degree 8. The image column gives the index of τ with respect to the numbering in the first column. All polynomials of the form $1+x+\ldots+x^8$ have depth 257.

3 Gandy Machines

In his 1936 paper [24], Turing gave cogent arguments for the thesis that anything that can be calculated by a human being in a routine fashion is computable on a Turing machine. Since his key application was the Entscheidungsproblem, there was no need to speculate about the possibilities of computation by general mechanisms. Furthermore, it was already observed by Church that any computational device with finite configurations and a recursive next-configuration function computes a recursive function, so that indeed Turing computability appears to be a natural environment.

In 1980 Gandy developed a general framework that appears to capture any reasonable notion of mechanism or device capable of synchronous, parallel computation. Gandy was motivated by Neumann's "crystalline automata", and space-time computation can be easily modeled in Gandy's framework. The devices conform to basic principles of physics, albeit not Newtonian physics: there is no instantaneous action at a distance. The main result in Gandy's paper is that, with very mild conditions on the way the parallel device is organized, one has the following theorem.

Theorem 3.1 *Any function computable on a Gandy Machine is recursive.*

Gandy gives a long list of counterexamples that demonstrate that his conditions cannot be further relaxed: any conceivable modification leads to machines that "display free will", i.e., that can compute any number theoretic function.

The main source of interest in Gandy Machines from the point of view of cellular automata is that they encapsulate precisely the basic principles of cellular automata: *local causation* and *unique reassembly*. Here, local causation means that the computation of $F(x)$, where F is the global map of the Gandy Machine, requires only local information about x, produces local pieces of information about $F(x)$, from which pieces $F(x)$ can be reassembled in a unique way. Moreover, Gandy Machines are in a sense coordinate-free, so that invariance under shift operations, and in fact under more general automorphisms of the underlying structure, is automatically satisfied.

Regrettably, Gandy's framework of structural maps on hereditarily finite sets with urelements is technically somewhat complicated, see the comments in [17] but also [18]. We will give a brief description of Gandy Machines in the next section, omitting some of the more tedious details. We caution the reader to consult the references for details. We then list some basic examples, and lastly study some of the complexity issues involved with Gandy Machines.

3.1 The Framework

A Gandy Machine is a discrete dynamical system $\langle \mathcal{C}, F \rangle$. The key idea is to divide the process of computing $F(x)$ into three phases. First, the input x is disassembled into a set of local approximations. A simple local rule is then applied to these pieces, and the resulting approximations for $F(x)$ are, in the last step, reassembled to produce $F(x)$.

$$
\begin{array}{cl}
x & \\
\downarrow & \text{disassembly} \\
\{\, u \mid u \sqsubseteq x \,\} & \\
\downarrow & \text{local map } v = G(u) \\
\{\, v \mid v \sqsubseteq F(x) \,\} & \\
\downarrow & \text{reassembly} \\
F(x) &
\end{array}
$$

It is clear that cellular automata fit this basic pattern precisely. Since Gandy is trying to capture as wide a class of mechanisms as possible, the configurations have to be fairly general combinatorial structures. To express the notion of location in some abstract way, the structures will contain *urelements* (or *atoms*), which are indistinguishable safe for equality. More precisely, fix a countable set of U urelements and define the hereditarily finite sets over U as follows.

$$
\begin{aligned}
\mathrm{HF}_0(U) &= \emptyset \\
\mathrm{HF}_{n+1}(U) &= \mathfrak{P}_\omega(U \cup \mathrm{HF}_n(U)) \\
\mathrm{HF}(U) &= \bigcup \mathrm{HF}_n(U)
\end{aligned}
$$

Here $\mathfrak{P}_\omega$ denotes the power set operation restricted to finite sets. Gandy's original definition excludes pure sets other than $\emptyset$, but that is a minor technical detail. The configurations of a Gandy Machine are elements of $\mathrm{HF}(U)$: $\mathcal{C} \subseteq \mathrm{HF}(U)$, and $F : \mathcal{C} \to \mathcal{C}$.

In order to hide information in the atoms, consider an automorphism π of $\mathrm{HF}(U)$. π is determined by a permutation on U. We say that x is *isomorphic to y over $A \subseteq U$*, $x \cong_A y$, iff $\exists \pi (A \subseteq Fix(\pi)$ and $y = x^\pi)$, and $x \cong y$ iff $x \cong_\emptyset y$. Write $x \cong_z y$ for $x \cong_{\mathrm{spt}\, z} y$ where $\mathrm{spt}\, z = U \cap TC(z)$ denotes the support of z, TC being the transitive closure operation. The *stereotype* of x is the collection of all isomorphic sets.

- Principle I: Form of Description.
  $\mathcal{C}$ and F are both *structural*:
  - $x \in \mathcal{C}$, $y \cong x$ implies $y \in \mathcal{C}$, and
  - $F(x^\pi) \cong_{x^\pi} F(x)^\pi$.

The last condition is stronger than $x \cong y \Rightarrow F(x) \cong F(y)$ but weaker than full invariance $F(x^\pi) \cong F(x)^\pi$. The idea is that $F(x)$ may contain new urelements, but that the old ones should persist. In practice, it is more convenient to use a finite set $U_0 \subseteq U$ of fixed atoms and only consider automorphisms over U_0. The next requirement restricts configurations to be of bounded rank, thereby limiting the combinatorial structures that can appear in a configuration.

- Principle II: Bounded Rank.
  We must have $\mathcal{C} \subseteq \mathrm{HF}_k(U)$ for some fixed k.

To approximate $x \in \mathrm{HF}(U)$ consider $P \subseteq TC(x)$, a so-called *set of parts (SoP)* for x. Define the *restriction* $x \downarrow P$ of x to P by

$$(x \cap P) \cup \bigcup \{\, z \downarrow (P \cap TC(z)) \;\big|\; z \in x \,\}.$$

u is an *approximation* to x, $u \sqsubseteq x$ iff $\exists P \subseteq TC(x)(u = x \downarrow P)$. Lastly, an *assembly* for x is a set of approximations. We say that x can be *uniquely assembled* from an assembly C iff C is not an assembly for any $y \neq x$.

Example:

$$x = \{a, b, c, \{a, b\}, \{a, c\}\}$$
$$x \downarrow \{a\} = \{a, \{a\}\}$$

Thus, some parts of the $\in$-tree of x are erased, and the result is properly collapsed to produce a set. Since the local map should just be a table lookup, we have to rule out unboundedly large approximations.

- Principle III: Bounded Assembly.
  There is a bound β such that for all $x \in \mathcal{C}$ there are SoPs $P_1, \ldots, P_m$ such that $|\mathrm{spt}\, P_i| \leq \beta$ and x can be uniquely reassembled from $\{\, x \downarrow P_i \;\big|\; i = 1, \ldots, m \,\}$.

Since x has bounded rank, there are only finitely many possible sets of parts up to isomorphism. Thus, disregarding urelements, there are only finitely many approximations. More precisely, there is a collection $\mathcal{T}$ of stereotypes so that any $x \downarrow P$ belongs to $\mathcal{T}$. We can now define the *causal neighborhoods* of x and the *determined regions* in $F(x)$, as produced by applying a local map G to the causal regions.

$$\mathrm{CN}(x) = \{\, u \sqsubseteq x \;\big|\; u \text{ maximal in } \mathcal{T} \,\}$$
$$\mathrm{DR}(x) = \{\, G(u) \;\big|\; u \in \mathrm{CN}(x), \mathrm{spt}\, G(u) \cap \mathrm{spt}\, x \subseteq \mathrm{spt}\, u \,\}$$

The map G is finite, up to isomorphism. The maximality condition simply serves to extract as large an amount of local information about x as possible. We require for all determined regions v and w that $v \cong_x w \Rightarrow v = w$. This leads us to the last condition a Gandy Machine has to satisfy.

- Principle IV: Local Causation (abridged version)
  $F(x)$ can be uniquely reassembled from $\mathrm{DR}(x)$.

To summarize, a *Gandy Machine* is a discrete dynamical system $\langle \mathcal{C}, F \rangle$ that satisfies Principles I through IV. A few examples are in order.

3.1.1 Cellular Automata

One-dimensional cellular automata are readily expressed in terms of Gandy Machines. The configurations here take the form

$$x = \{\langle p_1, s_1, p_2\rangle, \langle p_2, s_2, p_3\rangle, \ldots, \langle p_n, s_n, p_{n+1}\rangle\}$$

where the free atoms p_i label cells. Cell states are given by fixed atoms s_i, and proximity information is expressed in the combinatorial structure of the configurations. It is straightforward to design an appropriate global map, and to check that all Principles are indeed satisfied.

We actually have omitted a crucial problem in this presentation: since the local maps may introduce new atoms, care has to be taken that atoms introduced by different approximations u_1 and u_2 of x are properly matched up. This is not a problem for 1-dimensional automata, since the number of new atoms is bounded, but for 2-dimensional cellular automata the number of new atoms is in general unbounded. Consider Moore neighborhoods. As before, we can represent a configuration x as a set containing elements of the form

$$\langle p, q_1, \ldots, q_8, s\rangle$$

where p is the label of the center cell, the q_i are the labels of the four neighbors, and s is a fixed atom indicating the state of cell p. For a cell along the boundary of a configuration, some of the q_i will be a special fixed atom # indicating that there is no cell in that direction. When the configuration of the cellular automaton expands, the #'s will have to be replaced by new atoms, and the choice of new labels will have to be coordinated between adjacent boundary cells.

This can be done by requiring the existence of a second local map, which provides suitable patches. We refer the reader to [4, 18] for details. At any rate, it is quite straightforward to express cellular automata of arbitrary dimension, and on a variety of underlying grids, in terms of Gandy Machines.

3.1.2 Regular Languages

Here is an example of a more algebraic computation. Suppose we wish to check membership in a regular language $L \subseteq \Sigma^*$. Let $\langle S, \circ\rangle$ by the syntactic semigroup of $L \subseteq \Sigma^*$. Thus S is finite, and there is a homomorphism $h : \Sigma^* \to S$ such that

$$L = \bigcup \{\, h^{-1}(s) \;\big|\; s \in S_0 \,\} \text{ some } S_0 \subseteq S.$$

We choose configurations of the form

$$x = \{\, \langle p, s, q\rangle \;\big|\; p, q \text{ positions}, s \in S \,\}$$

where an input $w = w_1, \ldots, w_n$ is represented by a configuration of the form $x = \{\langle p_1, h(w_1), p_2\rangle, \langle p_2, h(w_2), p_3\rangle, \ldots\}$. The local map is given by

$$G(\{\langle p, s, q\rangle, \langle q, t, p'\rangle\} = \{\langle p, s \circ t, p'\rangle\}$$

Thus, regular language recognition on a Gandy Machine is in $O(\log n)$ time.

3.1.3 Circuits

More generally, one can simulate circuits in a natural fashion. AC^k is the collection of all log-space uniform circuits of depth $O(\log^k n)$ and polynomial number of gates, with unbounded fan-in. AC^k can be handled in $O(\log^k n)$ time on a Gandy Machine, so that NC is contained in poly-log time on a Gandy Machine.

Configurations here are essentially sets containing 4-tuples

$$\langle p, t, \{q_1, \ldots, q_n\}, s\rangle$$

where $p, q_1, \ldots, q_n$ are free atoms that label the gates of the circuit.

Furthermore, $t \in \{\vee, \wedge, \neg, I\}$, $s \in, \subseteq \{0, 1, \perp\}$ are composed of fixed atoms. The local map for or-gates has the form

$$\begin{aligned}
G(\{\langle p, \vee, \{q\}, \{\perp\}\rangle, \langle q, _, \emptyset, s\rangle\}) &= \{\langle p, \vee, \{q\}, TC(s)\rangle\} \\
G(\{\langle p, \vee, \{q\}, \{\perp, 0\}\rangle, \ldots\}) &= \{\langle p, \vee, \{q\}, \{\perp\}\rangle\} \\
G(\{\langle p, \vee, \{q\}, \{1, \ldots\}\rangle, \ldots\}) &= \{\langle p, \vee, \emptyset, 1\rangle\} \\
G(\{\langle p, \vee, \{q\}, \{0\}\rangle, \ldots\}) &= \{\langle p, \vee, \emptyset, 0\rangle\} \\
G(\{\langle p, \vee, \emptyset, s\rangle, \ldots\}) &= \{\langle p, \vee, \emptyset, s\rangle\}
\end{aligned}$$

The definitions for not/and-gates are similar. The ellipses indicate that the approximations obtained by restricting x to sets of parts are of the form $P = \{p, q, \vee, \wedge, \neg, I, 0, 1, \perp\}$ may contain other, irrelevant parts.

3.1.4 Parallel RAMs

It is natural to ask about the relationship between PRAMs and Gandy Machines. It was shown by Shepherdson [17] that one can implement a PRAM operating over a FOD structure $\mathcal{A} = \langle A, R_1, \ldots, R_r, f_1, \ldots, f_s\rangle$ on a Gandy Machine.

Theorem 3.2 *Shepherdson Computability on a Gandy Machine over a structure is equivalent to computability by a synchronous parallel procedure.*

See [17, 16] for details.

3.2 Computing $y = F(x)$ on a Gandy Machine

The previous examples show that Gandy Machine do in fact proved a fairly natural model for synchronous parallel computation. Repeated application of the global map allows one to emulate parallel computations in a very general sense. How about a single step in a Gandy Machine? In particular, considering the

framework of sets rather than ordered combinatorial structures, with a natural notion of proximity, how does local causation really work?

The condition that configurations are of bounded rank, and can be uniquely reassembled from parts of bounded size, imposes a fairly strong restriction on admissible configuration spaces. The key obstruction is the following. Consider

$$x = \{x_1, \ldots, x_n\}, x_i \subseteq U \text{ finite}$$

On can show that x can be reassembled for bound $\beta = n$, but cannot be reassembled for any $\beta < n$. Hence, for an admissible configuration space containing sets x as above, the cardinality n of x has to be bounded. Sets of arbitrary size cannot occur in a nested fashion. Note that the configurations in example 3.1.3 do fall prey to this obstruction.

Theorem 3.3 *Disassembly and application of the local map can be done in constant time on a PRAM.*

Proof sketch:
We can code hereditarily finite sets with urelements as finite FOD structures

$$\langle TC(x), \in, U, \ldots \rangle$$

More precisely, we consider structures of the form

$$\mathcal{A} = \langle \{0, \ldots, n-1\}, R_\in, R_U, \ldots \rangle$$

where $n = |TC(x)|$. E.g. $R_\in$ uses n^2 bits, R_U uses n bits, and so on. Note that this representation is not unique: we can enumerate the support of x arbitrarily.

Now suppose φ is a FOD formula of appropriate type. By a theorem of Immerman [6],

$$\mathcal{A} \models \varphi \text{ can be tested in constant time an a CRAM.}$$

Here a CRAM is a type of CRCW PRAM with priority writes, and a shift operation $\text{shift}(x, y) = \lfloor x/2^y \rfloor$. It is clear that $\mathcal{A} \models \varphi$ is clearly in $\text{SPACE}(\log n)$, parallelism gets us down to constant time. One can then show

Claim 1: $u \sqsubseteq x$ can be expressed as a FOD formula.
Claim 2: $v = G(u)$ can be expressed as a FOD formula.

Hence, using polynomially many processors, we can construct the set of approximations $\{ G(u) \mid u \sqsubseteq x, \text{ matches } \mathcal{T} \}$ for $F(x)$ in constant time. □

As one might suspect, reassembly is a bit more complicated. Note that we have to calculate a FOD structure representing $TC(F(x))$. Using a definability approach as in the last argument, it is not hard to see that one can test $y = F(x)$, given structures that code the transitive closures of x and y, in constant

time. Indeed, for all the natural examples from above, reassembly is possible in logarithmic time on a CRAM. The processors can agree on a coherent naming scheme for the new atoms in logarithmic time, and then assemble the sets in logarithmic time using a divide-and-conquer approach. A similar approach also works for configurations of the form mentioned as the key obstruction above. Since in a Gandy Machine sets of arbitrary size cannot occur in a nested fashion, one can generalize this method to arbitrary configurations.

Theorem 3.4 *Reassembly of $F(x)$ from approximations $\{ v \mid v \sqsubseteq F(x) \}$ can be done in logarithmic time on a CRAM.*

Thus, a single step in a Gandy Machine can be performed in a fairly low parallel complexity class. A similar result is claimed in [3], but we have been unable to verify their proofs.

We conclude with a comment about the nature of "local causation" in Gandy Machines. In the framework of sets over anonymous urelements, locality has to be taken with a grain of salt. To extract information about a neighboring component from a set, it seems necessary to admit something akin to unbounded fan-in in the world of circuits, or to increase the number of processors in the PRAM nonlinearly. Neither alternative is really adequate from the point of view of physical realization: a real gate cannot have an unbounded number of inputs, and a great many processors communicating with each other do not process information locally. By contrast, low-dimensional cellular automata admit perfectly reasonable physical models.

Problem: Find natural restrictions on Gandy Machines that are more consistent with local causation in physics. In particular, find machines that correspond more naturally to cellular automata.

One possible line of attack would be to impose some amount of structure on the set of urelements, and to permit the local maps to make use of this structure. For example, one might have an order structure on U. However, it is not clear how to do this without destroying the versatility of the model.

References

[1] J. T. Baldwin and S. Shelah. On the classifiability of cellular automata. To appear in TCS, 1999.

[2] R. Barua and S. Ramakrishnan. σ-game, σ^+-game, and two-dimensional cellular automata. *Theoretical Computer Science*, 154(2):349–366, 1996.

[3] E. Dahlhaus and J. A. Makowsky. *Gandy's Principles for Mechanisms as a Model for Parallel Computation*, pages 283–288. Volume 2 of *Computerkultur* [5], 2nd edition, 1994.

[4] R. O. Gandy. Church's thesis and principles of mechanisms. In J. Barwise et al., editor, *The Kleene Symposium*, pages 123–148, Amsterdam, 1980. North-Holland.

[5] R. Herken. *The Universal Turing Machine: A Half-Century Survey*, volume 2 of *Computerkultur*. Springer-Verlag, 2nd edition, 1994.

[6] N. Immerman. *Computational Complexity Theory*, volume 38 of *Proc. Symp. Appl. Math.*, chapter Descriptive and Computational Complexity, pages 75–91. AMS, 1989.

[7] D. S. Johnson. A catalogue of complexity classe. In J. van Leeuwen, editor, *Handbook of TCS*, volume A, chapter 2. Elsevier Science Pub., 1997.

[8] M. E. Lucas. Sur les congruences des nombres Euleriennes et des coefficients différentiels des functions trigonométrique, suivant un module premier. *Bull. Soc. Math. France*, 6:49–54, 1878.

[9] O. Martin, A. M. Odlyzko, and S. Wolfram. Algebraic properties of cellular automata. *Commun. Math. Phys.*, 93:219–258, 1984.

[10] C. Moore. Quasilinear cellular automata. *Physica D*, 103:100–132, 1997.

[11] C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[12] H. Rogers. *Theory of Recursive Functions and Effective Computability*. McGraw Hill, 1967.

[13] P. Sarkar. σ^+ automata on square grids. Technical Report ASC/96/9, Indian Statistical Institute, Calcutta, 1996.

[14] P. Sarkar and R. Barua. Multidimensional σ-automata, π-polynomials and generalized S-matrices. *TCS*, 197:111–138, 1998.

[15] J. C. Shepherdson. Machine configuration and word problems of given degree of unsolvability. *Zeitschrift f. Math. Logik u. Grundlagen d. Mathematik*, 11:149–175, 1965.

[16] J. C. Shepherdson. Computation over abstract structures: Serial and parallel procedures and Friedman's effective definitional schemes. In H. E. Rose at al., editor, *Logic Colloquium*, Amsterdam, 1973. North-Holland.

[17] J. C. Shepherdson. *Mechanisms for computing over abstract structures*, pages 537–556. Volume 2 of *Computerkultur* [5], 2nd edition, 1994.

[18] W. Sieg and J. Byrnes. An abstract model for parallel computation: Gandy's thesis. Technical Report CMU-PHIL-89, CMU, 1998.

[19] K. Sutner. σ-automata and Chebyshev polynomials. To appear in TCS.

[20] K. Sutner. On σ-automata. *Complex Systems*, 2(1):1–28, 1988.

[21] K. Sutner. De Bruijn graphs and linear cellular automata. *Complex Systems*, 5(1):19–30, 1991.

[22] K. Sutner. The complexity of finite cellular automata. *Journal of Computer and Systems Sciences*, 50(1):87–97, 1995.

[23] T. Toffoli and Margolus. Injective cellular automata. *Physica D*, 45(1–3):386–395, 1990.

[24] A. M. Turing. On computable numbers, with an application to the Entscheidungsproblem. *P. Lond. Math. Soc.*, 42:230–65, 1936.

[25] T. Yaku. The constructibility of a configuration in a cellular automaton. *Journal of Computer and Systems Sciences*, 7:481–496, 1973.