Discrete Mathematics and Theoretical Computer Science **AB(DMCS)**, 2003, 117–128

On undecidability of equicontinuity classification for cellular automata

Bruno Durand[1] and Enrico Formenti[1] and Georges Varouchas[1]

[1]*Laboratoire d'Informatique Fondamentale de Marseille (LIF),*
Centre de Mathématique et Informatique (CMI), 39 rue Joliot-Curie, 13453 Marseille Cedex 13, France.
email: {`bdurand,eforment,gvarouch`}`@cmi.univ-mrs.fr`

Equicontinuity classification is a popular classification of cellular automata based on their dynamical behavior. In this paper we prove that most of its classes are undecidable.

Keywords: cellular automata, classification, discrete dynamical systems, undecidability

Introduction

Cellular automata are simple model for the study of complex phenomena produced by simple local interactions. They consist of a regular lattice of *cells*. Each cell contain finite automaton which has a *state* chosen from a finite set of states. Updates are made according to a local rule which takes into account the current state of the cell and those of a fixed finite set of its neighboring cells. All cells are updated synchronously. A snapshot of the state of all cells at the same time is called a *configuration*.

Despite of their definition simplicity, cellular automata exhibit a wide range of dynamical behaviors. The classification of such behaviors is known as "the classification problem". It has captivated researchers for years and a complete solution does not appear to be on coming.

The first empirical classification was proposed about twenty years ago by Wolfram after an extensive experimental work. Successive studies of researchers in the field, tried to give this classification a formal justification. The first attempts of Čulik *et al.* [CY88, CPY89], made immediately clear were the problem is: CA behavior is so complex that almost any question about their long-term behavior is undecidable. Unfortunately we do not know how to formalize (and then prove) this idea. Two attempts in this direction are [Kar94] (a Rice theorem with *variable* states) and [CD00] (a *partial* Rice theorem with fixed states).

In the middle of nineties, P. Kůrka proposed to classify CA according to their degree of equicontinuity [Kur97]. Denote E the set of equicontinuity points of a CA and X the set of all configurations. Kůrka devised the following four classes

K_1) $E = X$;
K_2) $E \neq \emptyset$;
K_3) $E = \emptyset$;
K_4) expansive CA .

Since many of these classes are defined by properties that are not strictly related to limit sets, one might suspect that at least some of them are decidable. In this paper we prove that the first three classes are undecidable. These undecidability results are meaningful and have consequences on other decision problems (e.g. nilpotency). Decidability (or undecidability) of the fourth class is still open.

1 Cellular automata, discrete dynamical systems

Cellular automata are formally defined as quadruples $\langle d, Q, N, \delta \rangle$. The integer d is the *dimension* of the space the cellular automaton works on. The set Q is a finite set of *states* of cells. The *neighborhood* $N = (x_1, \ldots x_v)$ is a v-tuple of distinct vectors of $\mathbb{Z}^d$. The x_i's are the relative positions of the neighbor cells with respect to the cell, whose new state is being computed. The states of these neighbors are used to compute the new state of the center cell by the *local function* of the cellular automaton $\delta : Q^v \mapsto Q$.

A *configuration* is an application from $\mathbb{Z}^d$ to Q. Let $X = Q^{\mathbb{Z}^d}$ be the set of all configuration. The local function induces a *global function* on X as follows

$$\forall c \in Q^{\mathbb{Z}^d}, \forall i \in \mathbb{Z}^d, \mathcal{A}(c)_i = \delta(c_{i+x_1}, \ldots, c_{i+x_v}) \ .$$

A state q is called *quiescent* if $f(q, q, \ldots, q) = q$ (in the sequel we explain how our results are still valid when no quiescent state exists). More than one state can be quiescent, but only one of them is distinguished as the quiescent state. Denote X_F the set of *finite configurations*, i.e., those configurations that are almost everywhere equal to the quiescent state — non-quiescent in a finite number of cells.

Let us now recall some classical definitions from discrete dynamical systems theory.

A *dynamical system* (X, F) consists of a compact metric space X and F is a continuous function from X to itself. Denote d the metric on X and F^n the n-fold composition of F with itself.

We are interested in the study of dynamical systems in symbolic spaces. $Q^{\mathbb{Z}^d}$ is endowed with the product topology (also called Cantor topology) of a countable product of discrete spaces on Q. For the sake of simplicity, we will study cellular automata in dimension $k = 1$ or 2. The generalization of results to higher dimensions is straightforward for all the results in the present paper.

The metric d on $Q^{\mathbb{Z}^d}$ is defined as $\forall x, y \in Q^{\mathbb{Z}^d}$, $d(x, y) = 2^{-n}$ where $n = \inf\{i \in \mathbb{N}, x_i \neq y_i \text{ or } x_{-i} \neq y_{-i}\}$. This metric induces the *product* topology on $Q^{\mathbb{Z}^d}$.

Denote by $Q^\star$ the set of finite words on Q. For $w \in Q^\star$, $|w|$ denotes the length of w. For $t \in \mathbb{Z}, w \in Q^\star$, the sets

$$[w]_t = \left\{x \in Q^{\mathbb{Z}}, x_t = w_0, \ldots, x_{t+|w|} = w_{|w|}\right\}$$

are called *cylinders* and form a basis for the product topology on $Q^{\mathbb{Z}}$. The *shift map* σ is often used as a paradigmatic example of chaotic symbolic system. It is defined as $\forall c \in Q^{\mathbb{Z}}, \forall i \in \mathbb{Z},\ \sigma(c)_i = c_{i+1}$. Cellular automata are exactly those continuous maps from $Q^{\mathbb{Z}}$ to $Q^{\mathbb{Z}}$ that commute with the shift (i.e., $\mathcal{A} \circ \sigma = \sigma \circ \mathcal{A}$) [Hed69].

Kůrka's classes are defined by properties on the whole set of configurations. Using Proposition 1.1 one can express the properties defining some of Kůrka's classes in terms of behavior on finite patterns.

Proposition 1.1 ([Knu94]) *Let $\mathcal{A}$ be a cellular automaton. Then $\mathcal{A}$ is sensitive to initial conditions w.r.t. finite configurations if and only if $\mathcal{A}$ is sensitive to initial conditions.*

A point $x \in$ is *ultimately periodic* if there exists two integers $\bar{t}, p$ $(p > 0)$ such that implies that $F^{pt+\bar{t}+i}(x) = F^{\bar{t}+i}(x)$ for $i,t \in \mathbb{N}$. The minimal $\bar{t}, p$ with such a property are called, respectively, the *length of the transient* and the *period* of x. If $\bar{t} = 0$ then x is called *periodic*.

A point $x \in X$ is an *equicontinuity* point for F if for any $\varepsilon > 0$ there exists $\delta > 0$ such that for any $y \in X$, $d(x,y) \leq \delta$ implies that $\forall t \in \mathbb{N}$, $d(F^t(x), F^t(y)) < \varepsilon$. If X is perfect and all of its points are equicontinuity points then F is *equicontinuous*.

In other words, an equicontinuous systems has a very stable dynamic. Small errors in initial conditions will not grow during evolutions. Periodic systems are a trivial example of equicontinous systems, although not all equicontinuos systems are periodic. In the special case of CA we have the following result.

Lemma 1.2 ([BT00]) *A CA is equicontinuous if and only if it is ultimately periodic.*

The notion of *blocking word* is rather intuitive central for characterizing cellular automata dynamics in the presence of equicontinuity points as we can see in Proposition 1.3. It will be crucial in the proof of the main result of this section.

A word $V \in Q^\star$ is *blocking* if there exists an infinite sequence of words $(v_t)_{t\in\mathbb{N}}$ such that

1. for any $t \in \mathbb{N}$, $|v_t|$ is finite, odd and greater than or equal to r;
2. for any $c \in [V]_{-(|V|-1)/2}$ and any $t \in \mathbb{N}$, $A^t(c) \in [v_t]_{-(|v_t|-1)/2}$.

In other words V partitions the evolution diagram of A in two completely disconnected parts: all perturbations made in one side are completely “blocked” by V.

Proposition 1.3 ([BT00]) *Any equicontinuity point has an occurrence of a blocking word. Conversely, if there exist blocking words, then any point with infinitely many occurrences of a blocking word to the left and to the right (of* 0*) is an equicontinuity point.*

Regularity is another property connected with periodicity. A system is regular if it has a dense set of periodic points.

A dynamical system (X,F) is *sensitive to initial conditions* if there exists $\varepsilon > 0$ such that for any $x \in X$ and any $\delta > 0$, there exists $y \in X$ such that $0 < d(x,y) < \delta$ implies that there exists $t \in \mathbb{N}$ such that $d(F^t(x), F^t(y)) \geq \varepsilon$. In other words a systems that is sensitive to initial conditions defeat all numerical simulations. In fact small error in the measure of initial configurations are magnified during evolutions at a point that after some iterations the “real evolution” and the simulated one can be quite different.

Remark 1.4 *In a perfect space, any system with no equicontinuity points is sensitive to initial conditions and vice-versa. This property, de facto, inspired the definition of Kůrka’s classes and separates class* $K_1 \cup K_2$ *from* K_3.

Expansivity is a stronger version of instability than sensibility to initial conditions (at least in perfect spaces). The system (X,F) is *expansive* if there exists $\varepsilon > 0$ such that for any $x,y \in X$, $d(x,y) > 0$ implies that $\exists t \in \mathbb{N}$ such that $d(F^t(x), F^t(y)) \geq \varepsilon$. Be the definitions, it is easy to see that K_4 is contained in K_3.

2 Undecidability of the class K_1

Theorem 2.1 is the main result of this section. Its proof adresses the following problem in Kari’s work [Kar94]: the set of states is *not* fixed. The hardest part of the proof consists in proving that, in our particular case, letting the set of states vary is not necessary.

Theorem 2.1 *Consider K_1 for a fixed set of states Q and a fixed dimension $d \in \mathbb{N}$. Then membership in K_1 is undecidable.*

Remark 2.2 *In [CY88], Culik and Yu prove that membership in K_1 is undecidable for one-dimensional CA for any fixed set of states when considering finite configurations. This result cannot be used here since there are CA that are equicontinuous over finite configurations but not on more general configurations. Consider, for example, the one-dimensional CA on $\{0,1\}$ with local rule $\delta(x,y,z) = xy$ for $x,y,z \in \{0,1\}$.*

In the proof of the main result of the section we make a reduction to a well-known undecidable problem for CA. To this purpose we need some more definitions and some results from literature.

A CA $\langle 1, Q, N, \delta \rangle$ is *nilpotent* if there exists an integer n such that $\forall x \in Q^{\mathbb{Z}^d}$, $F^n(x) = \underline{c}$ for a quiescent configuration $\underline{c}$. Given a CA $\langle 1, Q, N, \delta \rangle$, a state $q \in Q$ is *spreading* if $\delta(x_1 \ldots x_{2r+1}) = q$ whenever there is at least one i $(1 \leq i \leq 2r+1)$ such that $x_i = q$. Even if nilpotent CA (even with a spreading state) have a quite simple dynamics, these properties are undecidable as proved by the following result.

As pointed out by J. Kari and N. Ollinger [KO03], the proof technique of [Kar94] allows to assert the following result (stronger form than published in [Kar94] because of restriction to CA with a spreading states).

Theorem 2.3 *Nilpotency is undecidable in the class of 1D cellular automata with a spreading state.*

However, Kůrka's classes are defined for a fixed set of states, while the proof of the previous result does not allow to bound the cardinality of the set of states. We prove a slightly stronger version of this result:

Proposition 2.4 *Nilpotency is undecidable for one-dimensional CA with state set $\{0,1\}$.*

Proof. Let $\mathcal{CA}_{01S}$ be the class of one-dimensional CA with state set $\{0,1\}$ such that $\delta(x_{-r} \ldots x_r) = 0$ if 0^r is contained in $x_{-r} \ldots x_r$. For any configuration c, call *segment* $w_{i,j}$ the word $w_{i,k} = c_{i+k}$ for $i \leq j$. Given a configuration c, we will say that a segment $w_{i,n}$ is *admissible* if it does not contain the word 1^n and $w_{i-n,2n}$ does not contain $010^{n-1}01$. Moreover we say that c is *n-admissible* if any of its segment of length n is admissible.

We reduce nilpotency to the problem of deciding nilpotency for CA in $\mathcal{CA}_{01S}$.

For any CA $\mathcal{A} = \langle 1, Q, \{-r, \ldots, r\}, \delta \rangle$ with a spreading state, build the CA

$$\mathcal{B}_{\mathcal{A}} = \langle 1, \{0,1\}, \{-rn, \ldots, rn\}, \delta_{\mathcal{A}} \rangle$$

where n is the cardinality of Q. Let $I_Q = 0, 1, \ldots, n-1$ be a relabeling of the states of $\mathcal{A}$ such that 0 is a spreading state. We will code any state $q \in I_Q$ by $\pi(q) = 01^q 0^{n-1-q}$.

For any n-admissible configuration c, for any $i \in \mathbb{Z}$, denote $q(i)$ the state of the simulate cell. Remark that $q(i)$ can always be detected from i; in fact it suffices to find the first occurrence of 01 to the left of i. Now define:

$$\delta_{\mathcal{A}}(c_{i-rn}, \ldots, c_{i+rn}) = \pi(\delta(q(i-r), \ldots, q(i+r))) \ . \tag{1}$$

By the above definition $\forall c \in Q^{\mathbb{Z}}$, $\mathcal{B}_{\mathcal{A}}(\pi(x)) = \pi(F(x))$ where F is the global function of $\mathcal{A}$. For all non $(2rn+1)$-admissible segments, let

$$\delta_{\mathcal{A}}(c_{i-rn}, \ldots, c_{i+rn}) = 0 \ . \tag{2}$$

Remark that since $\mathcal{A}$ is spreading then, by (2), $\mathcal{B}_{\mathcal{A}}$ is $\mathcal{CA}_{01S}$. We claim that $\mathcal{A}$ is nilpotent if and only if $\mathcal{B}_{\mathcal{A}}$ is nilpotent.

Assume that $\mathcal{A}$ is nilpotent. Then there exists $n \in \mathbb{N}$ such that $F^n(c) = \underline{0}$. For all n-admissible configurations, by (1) and (2), we have that $F^n_{\mathcal{B}}(c) = \underline{0}$. If c is not n-admissible, then let $i \in \mathbb{Z}$ be the index of a cell in a non-admissible segment. Remark that 0^n will occur in $F_{\mathcal{B}}(c)_{i-rn,2rn}$. Then since $\mathcal{B}_{\mathcal{A}}$ is in $\mathcal{CA}_{01S}$ one finds $F^n_{\mathcal{B}}(c)_i = 0$.

Assume that $\mathcal{A}$ is not nilpotent. By definition, there exists c such for all $n \in \mathbb{N}, F^n(c) \neq \underline{0}$. Therefore, by (1), one finds that $\forall n \in \mathbb{N}, F^n_{\mathcal{B}}(\pi(c)) = \pi(F^n(c)) \neq \underline{0}$. We thus get the required reduction. □

The previous result can be generalized to arbitrary dimension and state set because 1-dimensional CA over $\{0,1\}$ are a decidable class among d-dimensional CA over Q ($\{0,1\} \subset Q$), and analogously nilpotent cellular automata are decidable class among the class of ultimately periodic cellular automata.

Corollary 2.5 *Nilpotency is undecidable for cellular automata on any fixed state set and dimension.*

Theorem 2.6 *Ultimate periodicity is undecidable among for CA on any fixed state set and dimension.*

Proof of Theorem 2.1. Use Theorem 2.6 and Lemma 1.2. □

3 Undecidability of the class K_2

Theorem 3.1 *For a fixed dimension d and alphabet Q, it is undecidable if a CA on $Q^{\mathbb{Z}^d}$ has a blocking word.*

We shall prove this result by proposing a reduction, inspired from [Sut89], of the classical Halting theorem to our problem. We will present this reduction in several steps: the first step describes a universal class of Turing machines with constrains on their global behaviour, the second step will be the reduction of this particular universal model to cellular automata, and the final step will prove that this last reduction can be done with a fixed number of states.

By Proposition 1.3, the previous theorem and Remark 1.4, we get the following theorems.

Theorem 3.2 *Given a set of states Q and a dimension d, the class $K_1 \cup K_2$ is undecidable on the class of CA over $Q^{\mathbb{Z}^d}$.*

Theorem 3.3 *Given a set of states Q and a dimension d, the class K_3 is undecidable on the class of CA over $Q^{\mathbb{Z}^d}$.*

The reduction we propose is a reduction of the Halting problem on Turing machines to our K_2 problem over one-dimensional cellular automaton. The main difficulty is that we want to enforce a fixed behavior of our cellular automaton on *all* configurations, but not all configurations correspond to admissible instances.

There is a classical way to simulate a Turing machine $\mathcal{M}$ with a CA $C = \langle 1, Q_C, r = 2, \delta \rangle$: let Q and Σ denote the set of states and the working alphabet of $\mathcal{M}$, and μ its transition rule. We set $Q_C = Q \cup \Sigma$, and we define δ as follows ($*$ matches any state, and σ any state in Σ):

- if $\mu(q_1,\sigma_1) = (q_2,\sigma_2,+1)$, then $\delta(*,*,q_1,\sigma_1,*) = \sigma_2$ and $\delta(*,q_1,\sigma_1,*,*) = q_2$;
- if $\mu(q_1,\sigma_1) = (q_2,\sigma_2,0)$, then $\delta(*,q_1,\sigma_1,*,*) = \sigma_2$ and $\delta(*,*,q_1,\sigma_1,*) = q_2$;
- if $\mu(q_1,\sigma_1) = (q_2,\sigma_2,-1)$, then $\delta(*,q_1,\sigma_1,*,*) = \sigma_2$, $\delta(*,\sigma,q_1,\sigma_1,*) = \sigma$ and $\delta(*,*,\sigma,q_1,\sigma_1) = q_2$;
- for any unmatched neighborhood, δ does not modify the central cell.

To represent the configuration of $\mathcal{M}$ in the state q, with its reading head at position i on the data string w, one can use the following finite configuration $^{\omega}0w_{[0..i-1]}qw_{[i..|w|]}0^{\omega}$. One evolution step of C on the above configuration simulates one computation step of $\mathcal{M}$ on w.

Remark that this simulation does not allow us to link the properties of $\mathcal{M}$ with a behavior of C on all its configuration: the simulation is not consistent with $\mathcal{M}$ when C acts on a configuration which does *not* represent a configuration of $\mathcal{M}$ (e.g. a configuration with several reading heads, or an infinite configuration *etc.*).

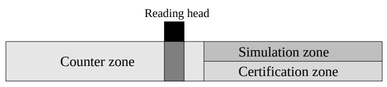
A solution to this problem is suggested by Sutner in [Sut89]. He proposes a simulation process which constrains the behavior of the automaton on all finite configurations, and which we enhance to ensure properties over all possible configurations.

Instead of directly simulating the computation of the machine, we build an automaton $\mathcal{C}_{\mathcal{M}}$ which simulates a machine $\mathcal{P}_{\mathcal{M}}$ working on the following set of predicates: $\{P_n :=$"after n computation steps starting on the empty entry ε, the machine $\mathcal{M}$ reaches the configuration w_n"$, n \in \mathbb{N}\}$, where $w_n = \mathcal{M}^n(\varepsilon)$.

The first step of our reduction will be the description of this machine $\mathcal{P}_{\mathcal{M}}$.

We represent P_n with the string $[\overline{n}|w_n]$ ($\overline{n}$ is the string representing n in unary). On this representation, $\mathcal{P}_{\mathcal{M}}$ will be able to perform two things: compute P_{n+1} from P_n (i.e., simulate $\mathcal{M}$'s computation), and also decide whether a given string represents a valid predicate or not (the set $\{P_n | n \in \mathbb{N}\}$ is recursive). In order to validate the behavior of $\mathcal{C}_{\mathcal{M}}$, we need to describe more precisely the behavior of $\mathcal{P}_{\mathcal{M}}$.

The data string of $\mathcal{P}_{\mathcal{M}}$ will have the following structure:



To implement such a structure, we use the alphabet $\Sigma_{\mathcal{P}}$ consisting in the union of the following sets:

- a singleton containing the blank character $B_{\mathcal{P}}$;
- a set Σ_{count} of two characters for the *counter* zone: $\boxed{\begin{matrix}1\\ \square\end{matrix}}$ and $\boxed{\begin{matrix}1\\ \boxtimes\end{matrix}}$ (both will be used to write k in unary, and we need to be able to "check" each bit during the certification cycle);
- a set Σ_{simul} of characters for the *simulation* and *certification* zone:

$$\Sigma_{simul} := \left\{ \boxed{\begin{matrix}a\\ b\end{matrix}} \middle| a, b \in \Sigma \cup Q \right\}$$

  (note that Σ also contains the blank character of the machine $\mathcal{M}$). We shall distinguish two subsets: the set $\Sigma_{char} := \Sigma \times (\Sigma \cup Q)$ (simulation of a character of $\mathcal{M}$) and the set $\Sigma_{head} := Q \times (\Sigma \cup Q)$ (simulation of the reading head of $\mathcal{M}$);
- a set of separators, in order to point clearly out the various zones in the data string: $[$,$|$ and $]$.

Call *ID* (as in Instantaneous Description) the set of syntactically valid data strings for our machine $\mathcal{P}_{\mathcal{M}}$:

$$ID := [\Sigma^*_{count} | \, \Sigma^*_{char} \Sigma_{head} \Sigma^*_{char}]$$

The computation of $\mathcal{P}_{\mathcal{M}}$ simulating $\mathcal{M}$ on the empty entry proceeds as follows:

1. Check if the data string is in *ID*;

2. simulate one computation step in the simulation zone (if necessary, extend the simulation zone by shifting the rightmost separator one cell to its right);
3. increment the counter by adding one extra $\boxed{\begin{matrix}1\\ \square\end{matrix}}$ to its left (thus extending the counter zone by one cell to its left);
4. uncheck the counter zone (i.e., replace all $\boxed{\begin{matrix}1\\ \boxtimes\end{matrix}}$ by $\boxed{\begin{matrix}1\\ \square\end{matrix}}$);
5. initialize the certification zone (it should just contain the string describing $\mathcal{M}$ in its starting configuration);
6. resimulate the k first steps of computation in the certification zone, checking one bit in the counter zone at each step;
7. compare the result obtained in the certification zone with the content of the simulation zone;
8. if no error occurred during the certification, check whether the simulated machine $\mathcal{M}$ has reached a halting configuration: if so, end the simulation (and enter an acceptance state), else proceed with the simulation loop.

During the simulation loop, several kind of errors can occur:

- during step 1, $\mathcal{P}_\mathcal{M}$ can detect a syntax error;
- no error can occur during the steps 2 to 5;
- during step 6, the certification process should not use more space than delimited by the separators. The certification can thus fail at this step if it appears that the space between the central and the rightmost separators is not enough;
- during step 7, $\mathcal{P}_\mathcal{M}$ can detect a simulation error, if the simulation does not match the certification;
- step 8 describes only the looping condition, and no error can appear here.

If any of these errors is raised, the machine enters a rejecting state, and halts.

Definition 3.4 (valid configuration) *A configuration of $\mathcal{P}_\mathcal{M}$ is* valid *when the computation of $\mathcal{P}_\mathcal{M}$ on this configuration does not lead to a rejecting state.*

We shall denote by $V_\mathcal{M}$ the set of valid configurations. $V_\mathcal{M}$ is recursive, and can be described as follows: a word w is in $V_\mathcal{M}$ if and only if

- $w \in ID$;
- if k represents w's counter and c its simulation zone, $c = \mathcal{M}^k(\varepsilon)$;
- the space between w's central separator and its rightmost separator is greater than or equal to the number of cells used by $\mathcal{M}$ during its k first computation steps.

The following remarks will be useful in the sequel.

Remark 3.5 *One whole simulation loop simulates exactly one computation step of $\mathcal{M}$.*

Remark 3.6 *By the certification process it holds that: $\mathcal{P}_\mathcal{M}$ will complete a simulation loop without entering a rejecting state if and only if its data string represents a predicate in $V_\mathcal{M}$.*

Remark 3.7 *Only Step* 2 *and* 3 *allow $\mathcal{P}_\mathcal{M}$ to use extra cells on its data string. Thus, if $\mathcal{P}_\mathcal{M}$ completes a simulation loop, the length of the data string increase by at most* 1 *to its right, and exactly* 1 *to its left.*

Denote by $Q_{\mathcal{P}}$ the set of states of $\mathcal{P}_{\mathcal{M}}$.

Lemma 3.8 *The three following statements hold:*

1. *$\mathcal{M}$ halts on the empty entry if and only if $\mathcal{P}_{\mathcal{M}}$ halts on* all *entries.*
2. *If $\mathcal{M}$ halts in k steps on ε, then, on any entry of size s, $\mathcal{P}_{\mathcal{M}}$ halts and uses at most $s+2k$ cells during its computation.*
3. *If $\mathcal{M}$ doesn't halt on ε, then there is an entry of $\mathcal{P}_{\mathcal{M}}$ on which $\mathcal{P}_{\mathcal{M}}$ doesn't halt and uses an unbound number of cells during its computation.*

Proof. Consider an entry w of $\mathcal{P}_{\mathcal{M}}$.

1. If $w \notin V_{\mathcal{M}}$, then by Remark 3.6, we know that $\mathcal{P}_{\mathcal{M}}$ halts on w.
   If $w \in V_{\mathcal{M}}$ then
   - either $\mathcal{M}$ halts on ε, then any computation on a configuration which derives from ε leads to termination, hence the computation of $\mathcal{P}_{\mathcal{M}}$ on w eventually terminates;
   - or $\mathcal{M}$ does not halt on ε; then, by Remark 3.5, $\mathcal{P}_{\mathcal{M}}$ will compute indefinitely many simulation loops on w, and will not stop.
2. By Remark 3.5, if $\mathcal{M}$ halts on ε in k steps, then any computation of $\mathcal{P}_{\mathcal{M}}$ involves at most k simulation loops, and therefore, by Remark 3.7, the growth of the data string is bounded by $2k$.
3. If $\mathcal{M}$ does not halt on ε, then, by Remark 3.5, $\mathcal{P}_{\mathcal{M}}$'s computation on $\left[\left|\boxed{\begin{matrix} q_0 \\ \_ \end{matrix}}\right|\right]$ does not halt too. The simulation loop is thus executed an unbounded number of times, and by Remark 3.7, one finds that after each simulation loop, the data string grows by at least one cell.

□

Let us now describe the second step of our reduction: $\mathcal{C}_{\mathcal{M}}$ will mainly consist in a classical simulation of $\mathcal{P}_{\mathcal{M}}$, but it will be able to erase the zones of its configurations which do not correspond to a configuration of $\mathcal{P}_{\mathcal{M}}$.

More precisely: we set $Q_{\mathcal{C}_{\mathcal{M}}} := \Sigma_{\mathcal{P}} \cup Q_{\mathcal{P}} \cup \{\texttt{kill}\}$. These states will play different roles in the sequel;

- 0 denotes the blank character $B_{\mathcal{P}}$;
- Q_{inert} denotes the subset $\Sigma_{\mathcal{P}} - \{B_{\mathcal{P}}\}$;
- Q_{head} denotes the subset $Q_{\mathcal{P}}$.

The radius is 4, and the local rule acts as follows:

1. if a cell is in the state `kill`, it enters the state 0;
2. if a cell's state is neither `kill` nor 0, and there is at least a cell `kill` state in its neighborhood, it enters the `kill` state;
3. if a cell's state is neither `kill` nor 0, and there is at a *local configuration error* in its neighborhood, it enters the `kill` state;
4. otherwise, the usual simulation rule of a Turing machine by a cellular automaton is applied.

A *local configuration error* on a given configuration is a local block (of radius 4) of the configuration which should not appear in a valid simulation. More precisely, we will have a local configuration error when the block contains:

- $\mathcal{P}_{\mathcal{M}}$'s rejecting state,
- non blank characters to the right of a right separator],
- non blank characters to the left of a left separator [,
- two reading heads, that is two cells whose states are in Q_{head}.

Definition 3.9 (data zone, active signal) *Given a configuration c, a* data zone *on this configuration will be any maximal connected part of c which does not contain* 0*.*

With the local rule of $\mathcal{C}_{\mathcal{M}}$ defined above, we can see that the only possible modifications can occur around cells with a state in $Q_{head} \cup \{\texttt{kill}\}$. We will call these states active signals*.*

A confi guration of $\mathcal{C}_{\mathcal{M}}$ can be seen as a sequence of data zones, on which $\mathcal{C}_{\mathcal{M}}$ acts through these active signals. The `kill` signal is invoked whenever a simulation error is detected, and destroys the zone in which it appears; the other signals (which represent the trajectory of $\mathcal{P}_{\mathcal{M}}$'s reading head) modify the data zone, possibly extending it.

Lemma 3.10 *If the `kill` state appears inside a finite data zone, this data zone will be entirely erased after a finite number of steps.*

Proof. With the radius of our automaton, the `kill` signal is at least twice as fast as the other active signals: it can act on a neighborhood of radius 4, while the other signals, following the classical simulation rule, act on a neighborhood of radius 2. Thus, if a `kill` signal appears in a fi nite data zone, the other signals will not be able to compensate indefi nitely its destruction. □

Definition 3.11 (valid zone) *A data zone will be* valid *when it represents a real valid configuration of $\mathcal{P}_{\mathcal{M}}$, that is a word in $V_{\mathcal{M}}$, with an adequately placed reading head.*

In other words: a valid data zone is a fi nite zone, which represents a real simulation of $\mathcal{P}_{\mathcal{M}}$, in which no `kill` signal will be invoked, unless it interacts with other data zones on the confi guration.

Lemma 3.12 *The two following hold:*

1. *if a data zone is not valid, it will grow by at most* 2 *cells;*
2. *if a data zone is valid, it will grow, for each complete loop simulated by $\mathcal{C}_{\mathcal{M}}$ in this zone, by exactly one cell to the left and at most one to the right.*

Proof. A data zone can only grow under the action of a signal in Q_{head}. Hence, if the invalid data zone does not contain a reading head character, it will not grow.

Else, if it is not valid, at least one `kill` signal will be invoked before one complete loop of $\mathcal{P}_{\mathcal{M}}$ could be simulated by $\mathcal{C}_{\mathcal{M}}$ in this data zone. Steps 2 and 3 of $\mathcal{P}_{\mathcal{M}}$'s loop will therefore be simulated at most once each, and the data zone will grow by at most 2 cells.

Statement 2 follows from the simulation process of $\mathcal{P}_{\mathcal{M}}$ by $\mathcal{C}_{\mathcal{M}}$, and Remark 3.7. □

Lemma 3.13 *If $\mathcal{M}$ halts on the empty entry in k steps, then 0^{2k+5} is a blocking word of $\mathcal{C}_{\mathcal{M}}$.*

Proof. By Lemma 3.12 that, if $\mathcal{M}$ only computes k steps on ε, then the growth of any data zone is bounded by k cells to the left and k to the right.

If the word 0^{2k+5} is present in the confi guration, it defi nes (at least) two data zones (one to its left and one to its right), which won't grow more than k cells to either side along the evolution of $\mathcal{C}_{\mathcal{M}}$. Hence the fi ve central cells of the word 0^{2k+5} will never be affected, which ends the proof. □

Lemma 3.14 *If $\mathcal{M}$ does not halt on the empty entry, then no word of Q^* is blocking for $\mathcal{C}_\mathcal{M}$.*

Proof. Consider a word w of Q^*, we want to prove that we can always create around w a confi guration which disturbs any letter of w. By Lemma 3.12, any valid data zone will grow indefi nitely to its left, unless it interacts with another data zone.

Consider a word v representing a valid zone. If we let $\mathcal{C}_\mathcal{M}$ evolve on the confi guration $c_0 := {}^\omega 0wv0^\omega$, then exactly one of the following statements holds:

- all the data zones of w disappear completely before the evolution of v gets a chance to interact with it;
- v's evolution interacts with some data zone of w.

If the fi rst case holds, then w is not a blocking word: v's evolution can proceed indefi nitely, and will eventually reach any cell to its left.

If the second case holds: a `kill` signal is invoked after a fi nite number of steps, in the middle of a fi nite data zone (the concatenation of w's rightmost data zone and v's evolution), so by Lemma 3.10, this data zone will be erased in a fi nite number of steps n_1.

Now, consider the evolution of the confi guration $c_1 := {}^\omega 0wv0^{2n_1}v0^\omega$.

We know that the fi rst v will disappear along with w's rightmost data zone after n_1 steps. The space between the two v makes it thus impossible for the second to interact with the fi rst. Therefore examining the evolution of $\mathcal{C}_\mathcal{M}$ on this confi guration, exactly one of two statements holds:

- all the data zones of wv disappear completely before the evolution of the second v gets a chance to interact with it;
- the second v's evolution interacts with some data zone of w.

If the fi rst case holds, w is not a blocking word.

In the second case: another data zone of w will be destroyed, after a certain number of steps n_2. In this case, we can consider the confi guration $c_2 := {}^\omega 0wv0^{2n_1}v0^{2n_2}v0^\omega$.

By induction, we can thus build an arbitrarily long sequence of confi gurations (c_k), such that the confi guration c_k "kills" k data zones of w. Since w is fi nite, it only has a fi nite number of data zones, and eventually disappears entirely. □

Corollary 3.15 *$\mathcal{C}_\mathcal{M}$ has an equicontinuity point if and only if $\mathcal{M}$ halts on ε.*

Proof of Theorem 3.2. We will now describe the fi nal part of our reduction: we need to prove that the reduction $\mathcal{M} \to \mathcal{C}_\mathcal{M}$ can be done even if the set of states of $\mathcal{C}_\mathcal{M}$ is fi xed. As in the preceding section, we will build a similar reduction with a 2-state automaton.

From $\mathcal{C}_\mathcal{M}$ build the 2-state automaton $\widetilde{\mathcal{C}_\mathcal{M}}$ which simulates $\mathcal{C}_\mathcal{M}$ as follows: choose a fi xed indexation of $\mathcal{C}_\mathcal{M}$'s states $Q = \{q_0, \ldots, q_n\}$, imposing $q_0 = 0$, and represent q_i by the binary sequence 01^i0^{n-i}.

A state of $\mathcal{C}_\mathcal{M}$ is thus represented by a sequence of $n+1$ states of $\widetilde{\mathcal{C}_\mathcal{M}}$. The radius of $\mathcal{C}_\mathcal{M}$ is 4, so a radius of $5(n+1)$ for $\widetilde{\mathcal{C}_\mathcal{M}}$ is enough for a cell to determine both the state to which it belongs, and the state of its two simulated neighbors.

The beginning of a simulated state is represented by the sequence 01, so that any cell of a simulated state is able to determine if, locally, the simulated confi guration is valid (there must be exactly a multiple of $(n+1)$ cells between two consecutive occurrences of 01).

The local rule of $\widetilde{\mathcal{C}_{\mathcal{M}}}$ proceeds as follows: if the neighborhood of a cell contains an alignment error, the cell enters the state 0, else the cell evolves according to the simulated rule.

Both Lemma 3.13 and 3.14 can be adapted to $\widetilde{\mathcal{C}_{\mathcal{M}}}$: the simulation of 0^{2k+5}, that is $0^{(2k+5)(n+1)}$, is a blocking word for $\widetilde{\mathcal{C}_{\mathcal{M}}}$ if $\mathcal{M}$ halts on ε; and if $\mathcal{M}$ does not halt, then any simulated valid zone will also grow indefi nitely, allowing to perturb any given fi nite word. $\square$

Acknowledgements

The authors are very grateful to Jarkko Kari and Nicolas Ollinger for pointing out the result about nilpotent CA with a spreading state that was fundamental to extend our proof in Section 2 (originally designed for dimension at least 2) to all dimensions.

References

[BT00] F. Blanchard and P. Tisseur. Some properties of cellular automata with equicontinuity points. *Annales de l'Instute Henri Poincaré*, 36(5):562–582, 2000.

[CD00] J. Cervelle and B. Durand. Tilings: Recursivity and regularity. In *STACS 2000*, volume 1770 of *LNCS*, pages 491–501, 2000.

[CPY89] K. Culik, J. Pachl, and S. Yu. On the limit set of cellular automata. *SIAM Journal on Computing*, 18:831–842, 1989.

[CY88] K. Culik and S. Yu. Undecidability of cellular automata classifi cation schemes. *Complex Systems*, 2:177–190, 1988.

[Hed69] G. A. Hedlund. Endomorphism and automorphism of the shift dynamical system. *Mathematical System Theory*, 3:320–375, 1969.

[Kar94] J. Kari. Rice's theorem for the limit set of cellular automata. *Theoretical Computer Science*, 127:229–254, 1994.

[Knu94] C. Knudsen. Chaos without nonperiodicity. *American Mathematical Montly*, 101:563–565, 1994.

[KO03] J. Kari and N. Ollinger. Why proof of indecidability of nilpotency applies to cellular automata with a spreading state. Personal communication, 2003.

[Kur97] P. Kurka. Languages, equicontinuity and attractors in cellular automata. *Ergodic Theory & Dynamical Systems*, 17:417–433, 1997.

[Sut89] K. Sutner. A note on culik-yu classes. *Complex Systems*, 3:107–115, 1989.