Theoretical Computer Science 134 (1994) 387–401
Elsevier

Inversion of 2D cellular automata: some complexity results*

B. Durand

Laboratoire de l'Informatique du Parallélisme, Ecole Normale Supérieure de Lyon, 46 Allée d'Italie, F-69364 Lyon Cedex 07, France

Communicated by M. Nivat
Received January 1993
Revised October 1993

Abstract

Durand, B. Inversion of 2D cellular automata: some complexity results, Theoretical Computer Science 134 (1994) 387–401.

In this paper, we prove the co-NP-completeness of the following decision problem: "Given a two-dimensional cellular automaton $\mathscr{A}$ (even with Von Neumann neighborhood), is $\mathscr{A}$ injective when restricted to finite configurations not greater than its length?" In order to prove this result, we introduce two decision problems concerning, respectively, Turing machines and tilings that we prove NP-complete. Then, we present a transformation of problems concerning tilings into problems concerning cellular automata.

1. Introduction

Cellular automata (CA) are often used for modeling complex natural systems with many rudimentary cells interacting locally with each other. Possible evolutions of cellular automata have been extensively studied in order to analyze evolutions of such natural systems. Problems like bijectivity or surjectivity of CA are very basic because they correspond to physical notions: conservation of information (which corresponds to physical reversibility) or reachability of all states.

In 1962–1963, Moore and Myhill proved the so-called "garden of Eden" theorem which proves that surjectivity is equivalent to injectivity on finite configurations

Correspondence to: B. Durand, Laboratoire de l'Informatique du Parallélisme, Ecole Normale Supérieure de Lyon, 46 Allée d'Italie, F-69364 Lyon Cedex 07, France. Email: bdurand@lip.ens-lyon.fr.

*This work was partially supported by the Esprit Basic Research Action "Algebraic and Syntactic Methods in Computer Science" and by the PRC "Mathématique et Informatique".

SSDI 0304-3975(93)E0206-J

[9,8]. Richardson proved in 1972 [10] that if a CA realizes a bijective function, then there exists another CA called its inverse that realizes the inverse function. The same year, Amoroso and Patt proved that the reversibility (or the surjectivity) of one-dimensional CA is decidable [1]. One-dimensional CA work on a bi-infinite line of cells, two-dimensional CA work on a plane tiled with square cells, etc.

Recently, Jarkko Kari proved that the reversibility of two-dimensional CA fails to be decidable [6, 7]. An easy consequence of this result is that the inverse CA of a reversible CA cannot be found by algorithm: its size can be greater than any computable function of the size of the reversible CA. The proof of this result consists in transforming the tiling problem of the plane which has been proved undecidable in 1966 by Berger [2, 11] into the reversibility problem on an adequate family of CA.

The main goal of this paper is to prove that, if we restrain the field of action of the two-dimensional CA to finite configuration bounded in size, it is still "difficult" to prove that the CA is (or is not) reversible. As far as we know, it would be the first complexity result concerning a global property of 2D CA. We prove that the decision problem presented above belongs to the class of co-NP-hard problems or to the class of co-NP-complete problems if the sizes of the considered finite configurations are supposed bounded by the size of the representation of the considered CA. Both of these complexity classes are supposed to contain only intractable decision problems.

Our result also holds for k-dimensional CA where $k \geqslant 2$. But in the case of one-dimensional CA, the reversibility problem can be solved in polynomial time in the size of the transition table (see [12] for another point of view on this problem).

We prove our result by introducing a very adequate set of tiles having an ad hoc property. Kari [7] has proved his undecidability results for two-dimensional CA by introducing a very complicated set of tiles. We keep the ideas of his construction but the tile set we use is much simpler. With this construction, we reduce decision problems concerning tilings into decision problems concerning CA. We have already used this method in order to provide a simple proof for the undecidability of the surjectivity problem in [4] (improved version in [5]).

In the next section, we give the usual definitions of cellular automata, tilings, and present well-known theorems related to our topics. In the following section we prove our main complexity result: the problem that we call CA-FINITE-INJECTIVE is co-NP-complete. Our proof consists in a reduction of a problem concerning finite tilings that we call FINITE-TILING. We prove this problem to be NP-complete by a reduction of another problem concerning the minimal computing time of non-deterministic Turing machines: NDTM-TIME. The reduction of FINITE-TILING into CA-FINITE-INJECTIVE is not so simple as the previous one. Anyway, all technical aspects of the proof are contained in the construction of a special set of tiles which verifies rather simple specific properties.

2. Definitions and basic properties

2.1. Cellular automata

Cellular automata are formally defined as quadruplets (n, S, N, f):

- The integer n is the *dimension* of the space the CA will work on.
- S is a finite set called the set of *states*.
- The *neighborhood* N is a v-tuple of distinct vectors of $\mathbb{Z}^n$. For us, $N=\{x_1, \ldots, x_v\}$: the x_i's are the relative positions of the neighbor cells with respect to a given center cell. The states of these neighbors are used to compute the new state of the center cell.
- The *local function* of the cellular automaton $f: S^v \mapsto S$ gives the local transition rule.

A *configuration* is an application from $\mathbb{Z}^n$ to S. The set of all the configurations is $S^{\mathbb{Z}^n}$ on which the *global function* G of the cellular automaton is defined via f and N:

$$\forall c \in S^{\mathbb{Z}^n}, \quad \forall i \in \mathbb{Z}^n, \quad G(c)(i)=f(c(i+x_1), \ldots, c(i+x_v)).$$

Note that two distinct cellular automata do not differ by the definition of their global function G: they are only characterized by n, S, N and f.

In the following, we consider two-dimensional CA ($n=2$).

Sometimes, a state q for which $f(q, q, \ldots, q)=q$ is distinguished in S and is called a *quiescent* state. A *finite configuration* is an almost everywhere quiescent configuration. If there exist two integers i and j such that all nonquiescent cells of the configuration are located inside a square of size $i \times j$, then, we say that the size of the finite configuration is smaller than (or equal to) $i \times j$.

In order to prove complexity results, it is very important to define precisely what are the sizes of the instances. In this paper, we use the following convention explained below.

Size: *The size necessary to code a cellular automaton is* $s^v \cdot \log s + o(s^v \cdot \log s)$ *where* s *is its number of states and* v *the number of elements of its neighborhood.*

The size of a CA is exactly the sum of the size of its local transition function and of the size of its neighborhood. The local transition function is only a v-dimensional table, hence its size is $s^v \cdot \log s + o(s^v \cdot \log s)$. The size of the neighborhood is the size of the coding of the coordinates of each neighbor cell. We assume in the following that this last size is lower than the size of the transition table, more precisely, that $\forall x \in N, |x| \leqslant s^v$. If it were not the case, the neighbors of a cell would be very far from it, hence a single iteration of the CA would be intractable! If we refuse this hypothesis, the problem we present is proved co-NP-*hard* but may not be in co-NP.

2.2. Turing machines

There exist many different, although equivalent, definitions of deterministic Turing machines (DTM) and nondeterministic Turing machines (NDTM). We briefly present

below our definitions, and we discuss what could be considered as the size of a DTM and a NDTM.

In our formalism, the two kinds of machines differ only by their transition function. Both of them have a *single bi-infinite tape*, on which 0's and 1's can be read and written by a read–write head. The digit "0" is sometimes called the *blank* symbol. The computations start on an almost everywhere blank tape. The input is written on the nonblank part of the tape. DTM and NDTM both have a *finite set of states S*. The number of states of a Turing machine τ is denoted by $\sigma(\tau)$. One of the states of S is called *initial state* and is denoted by q_I. Two special states are added but do not belong to S: an *acceptance halt-state* q_Y and a *refusal halt state* q_N. These two models differ by the definition of their transition function:

- for a DTM, the transition function δ associates a new state, a new letter and a movement of the head to a state and a letter. Formally,

$$\delta : S \times \{0,1\} \mapsto (S \cup \{q_Y, q_N\}) \times \{0,1\} \times \{left, right\};$$

- for a NDTM, the transition function δ associates an arbitrary number of triplets of a possible new state, a new letter and a movement of the head to a state and a letter. Formally,

$$\delta : S \times \{0,1\} \mapsto \mathcal{P}(S \cup \{q_Y, q_N\} \times \{0,1\} \times \{left, right\}).$$

The computation of a Turing machine begins in the initial state, and with the head located on the leftmost letter of the input word of the machine. It consists in the iteration of the following "procedure":

(1) If the machine is in a halt-state, then the machine stops and says "yes" if in q_Y, "no" if in q_N. If the Turing machine is not only a decision machine but also computes an output, the output word is supposed to be located at the right of the current position of the head.

(2) If the machine is not in a halt-state, then let q be the current state, and x be the letter on the tape at the current position of the head.

- If the machine is deterministic, and if $\delta(q,x)=(q',x',d)$, then the machine writes x' on the tape with the head, enters the new state q' and its head moves according to d.
- If the machine is nondeterministic, and if $\delta(q,x) \ni (q',x',d)$, then the machine can write x' on the tape with the head, enter the state q', and move according to d.

In the following, our study only concerns decision problems. That is why we consider only decision Turing machines: when given an input, we expect them to answer "yes" or "no" if the computation ends, but what is written on its tape does not matter.

Lemma 1. *Consider a TM τ. The size necessary to code the machine is a polynomial function of its number of states $\sigma(\tau)$ even if τ is nondeterministic.*

Proof. First recall that the size necessary (and sufficient) to code a function $f: A \mapsto B$ is exactly $a \log b$ if a and b denote the number of elements of A and B, respectively. Hence the size of the transition function δ of τ is

- $2 \cdot \sigma(\tau) \cdot \log(4 \cdot (\sigma(\tau)+2))$ if τ is a DTM,
- $2 \cdot \sigma(\tau) \cdot \log(4 \cdot 2^{(\sigma(\tau)+2)}) = 2 \cdot \sigma(\tau) \cdot (\sigma(\tau)+4)$ if τ is a NDTM.

As the coding of the number of elements of the set of states requires $\log(\sigma(\tau))$ bits, then the size necessary to code the machine is a polynomial function of the number of states. □

2.3. Tilings

A tile is a square, the sides of which are colored. The colors belong to a finite set C called the *color set*. A set of tiles τ is a subset of C^4. All tiles have the same (unit) size. A tiling of the plane is *valid* if and only if all pairs of adjacent sides have the same color. Notice that it is not allowed to turn tiles. The following well-known theorem is due to Berger [2] in 1966 and a simplified proof was given in 1971 by Robinson [11].

Theorem 1. *Given a tile set, it is undecidable whether this tile set can be used to tile the plane.*

We can also define *finite tilings*. We assume that the set of colors contains a special "blank color" and that the set of tiles contains a "blank tile", i.e. a tile whose sides are blank. A finite tiling is an *almost everywhere blank* tiling of the plane. If there exist two integers i and j such that all the nonblank tiles of the tiling are located inside a square of size $i \times j$, then we say that the size of the finite tiling is lower than $i \times j$. Notice that inside the $i \times j$ square, there can be blank and nonblank tiles. If there is at least one nonblank tile, then the tiling is called *nontrivial*.

Another undecidability result can be proved simply by using a construction presented by Kari in [7] which reduces the undecidability of the halting problem for Turing machines into it.

Theorem 2. *Given a tile set with a blank tile, it is undecidable whether this tile set can be used to form a valid finite nontrivial tiling of the plane.*

3. Complexity results

In this section, we prove that the problem of deciding whether a given CA is reversible when restricted to finite configurations the size of which is lower than $n \times n$ (where n is the size of the CA) is co-NP-complete. We call this problem CA-FINITE-INJECTIVE. In order to prove this result, we introduce two decision problems that we prove to be NP-complete: NDTM-TIME and FINITE-TILING.

3.1. NDTM-TIME is NP-complete

NDTM-TIME:

Instance: A nondeterministic Turing machine. An integer n lower than the number of states of the machine.

Question: Is there a computation of the machine beginning on an empty tape and halting after less than n steps?

Proof. This problem belongs to NP: we construct a NDTM μ' that, given an instance of the problem (i.e. a NDTM that we call μ and an integer n lower than the number of states of μ), computes n transitions of μ and after these computations, if μ answers "yes" then μ' answers "yes", else, if the computation is not finished or if the answer of μ is "no" then μ' answers "no". All possible computations of μ can be simulated by μ'.

Consider an universal Turing machine as described for instance in [14]. This machine, when given as input a deterministic Turing machine γ, is able to compute the evolution of γ on the blank tape. The time needed to compute a step is lower than a polynomial function of the size of γ. We transform this universal DTM into a universal NDTM: when the input is a NDTM μ, then if μ offers a choice in its evolution, then the universal NDTM offers the same choice. Thus, the universal NDTM simulates an arbitrary computation of the NDTM μ in polynomial time.

μ' acts as follows: on the input n and μ, it computes one step of the universal NDTM with μ as input, then it substracts 1 to n, then computes another step, substracts 1 to $n-1$, etc. When $n=0$, then if the simulation of μ has ended on the acceptance halt-state q_Y, then the NDTM μ' halts in its acceptance state. Else it halts in its refusal state. As n is lower than the number of states of μ, then the computation time for each step so defined is polynomial. Hence there exists an acceptance computation of μ' on the entrance (μ, n) if and only if there exists a computation of the machine μ beginning on an empty tape and halting after less than n steps. NDTM-TIME is in NP.

To prove that this problem is NP-complete, we prove that any problem in NP polynomially reduces to NDTM-TIME. Our proof is much more simple than Cook's proof which proves that any problem in NP reduces to the well-known SATISFIABILITY problem [3]. Recall that a problem π is said to *polynomially reduce* to a problem π' if and only if there exists a polynomial-time (deterministic) algorithm which transforms any instance x of π into an instance x' of π'. It is also required that the answer of π on x should be "yes" if and only if the answer of π' on x' is "yes" [13].

Let π be any problem in NP. There exists a NDTM which solves π in polynomial time. Let μ_π be such a NDTM and R the associated polynomial function. We transform μ_π in the following way:

(1) We first construct a new NDTM μ'_π. We conserve all states and transitions of μ_π. We transform the halt-state q_N of μ_π into a "normal" new state q' of μ'_π. and add the following transitions:

$$\delta(q', 0) = \{(q', 0, r)\},$$

$$\delta(q', 1) = \{(q', 1, r)\}.$$

It means that, in μ'_π, if a computation enters the state q', then it is absolutely sure that the computation will not reach any halt-state and thus will never halt. We create a new refusal halt-state for μ'_π with no transitions arriving in it.

The new NDTM μ'_π halts on an input if and only if μ_π halts on the same input and answers "yes". μ'_π either halts and answers "yes" or does not halt.
The size of μ'_π is just a constant larger than the size of μ_π.

(2) Let x be an instance of the problem π. We construct a new NDTM $\mu_{(\pi, x)}$ such that $\mu_{(\pi, x)}$ writes x on its tape and then runs μ'_π (μ'_π reads x as input). If we run $\mu_{(\pi, x)}$ on a blank tape then the answer of $\mu_{(\pi, x)}$ is "yes" if and only if x is a solution for π. If x is not a solution for π then $\mu_{(\pi, x)}$ does not halt. Furthermore, the answer is never "yes" after the polynomial time $2|x| + R(|x|)$, where $|x|$ denotes the size of x.

The size of $\mu_{(\pi, x)}$ is a polynomial function of the size of μ_π and of the size of x. Its number of states is $|x| + \sigma(\mu_\pi) + 2$; remember that $\sigma(\mu_\pi)$ is the number of states of the Turing machine μ_π. Hence, given x and μ_π, $\mu_{(\pi, x)}$ can be computed by a polynomial-time algorithm.

(3) We add new states to those of $\mu_{(\pi, x)}$. As we intend that these added states do not modify any evolution of $\mu_{(\pi, x)}$, we do not add new transitions. The number of added states is exactly $R(|x|) + |x| - \sigma(\mu_\pi) - 2$ which is a polynomial function. This addition of states can be computed by a polynomial-time algorithm on the inputs x and μ_π. The new NDTM is called $\mu'_{(\pi, x)}$.

Given a problem π in NP and an input x of this problem, there exists a polynomial-time algorithm that computes the NDTM $\mu'_{(\pi, x)}$. The answer for π on the instance x is "yes" if and only if the answer for NDTM-TIME on the instance $\mu'_{(\pi, x)}$ is "yes". Hence we have proved that π is polynomially reducible to NDTM-TIME. □

3.2. *From NDTM-TIME to FINITE-TILING*

FINITE-TILING:
Instance: Finite set C of colors with a blank color, collection $\tau \subset C^4$ of tiles including a blank tile. A positive integer $n \leqslant |C|$.
Question: Is there a finite nontrivial tiling of the plane of size lower than $n \times n$?

Proof. First note that this problem trivially belongs to NP since it is polynomial to check whether a given tiling is valid.

Now consider an instance of NDTM-TIME, i.e. a NDTM μ and an integer n lower than the number of states $\sigma(\mu)$ of the machine. We first construct a set of tiles associated to μ. Let S be the set of states of μ. An almost identical construction can be found in [7]. The goal of this construction was to prove that the problem of knowing whether there exists a finite nontrivial tiling of the plane is undecidable. We define below the colors used by our tiles:

- A "blank" color B.
- Two "initialization" colors I and I'.
- A "halting" color H.

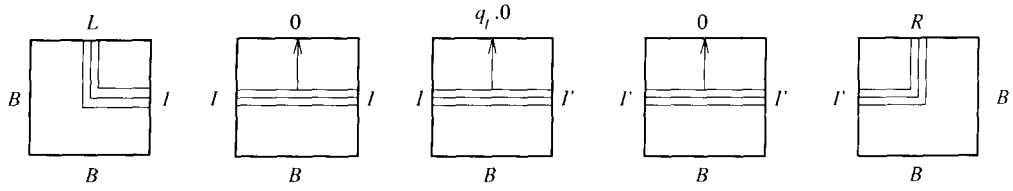

Fig. 1. The "initialization" tiles.

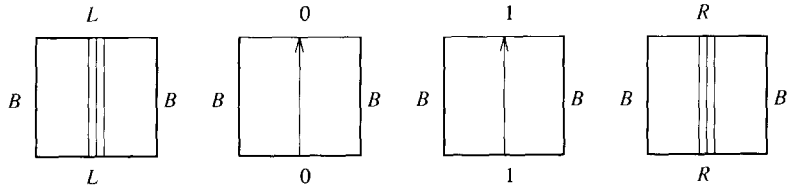

Fig. 2. The "static" tiles.

- A "left" color L.
- A "right" color R.
- A "state" color s for each $s \in S \cup \{q_Y, q_N\}$.
- A "symbol" color a for each $a \in \{0, 1\}$.
- A "state-symbol" color $s \cdot a$ for each $s \in S \cup \{q_Y, q_N\}$ and for each $a \in \{0, 1\}$.

With these colors, we construct a tile set τ: we specify the tiles by the color of their North, East, South, and West sides:

- The "blank" tile (B, B, B, B).
- The "initialization" tiles (Fig. 1)

$$(L, I, B, B), \quad (0, I, B, I), \quad (q_I \cdot 0, I', B, I), \quad (0, I', B, I'), \quad (R, B, B, I').$$

Remember that "0" is the blank symbol of μ and q_I its initial state.

- The "static" tiles (Fig. 2)

$$(L, B, L, B), \quad (0, B, 0, B), \quad (1, B, 1, B,), \quad (R, B, R, B).$$

- The "computation" tiles of 2 kinds (Fig. 3):
  - $(a', B, s \cdot a, s')$ if the transition $(s', a', \textit{left})$ is a *possible* image of (s, a).
  - $(a', s', s \cdot a, B)$ if the transition $(s', a', \textit{right})$ is a *possible* image of (s, a).
- The "merging" tiles of 2 kinds (Fig. 4):
  - $(s \cdot a, s, a, B)$ for each $s \in S \cup \{q_Y, q_N\}$ and for each $a \in \{0, 1\}$.
  - $(s \cdot a, B, a, s)$ for each $s \in S \cup \{q_Y, q_N\}$ and for each $a \in \{0, 1\}$.
- The "halting" tiles (Fig. 5)

$$(B, H, L, B), \quad (B, H, 0, H), \quad (B, H, 1, H,), \quad (B, H, s_h \cdot a, H), \quad (B, B, R, H).$$

for each $a \in \{0, 1\}$ and for each $s_h \in \{q_Y, q_N\}$.

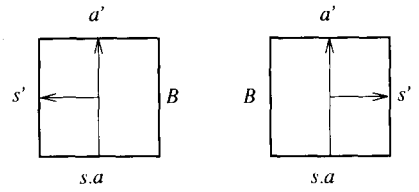

Fig. 3. The "computation" tiles.

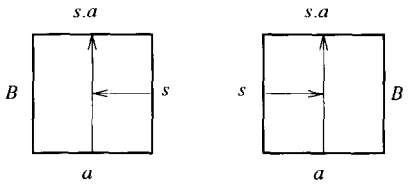

Fig. 4. The "merging" tiles.

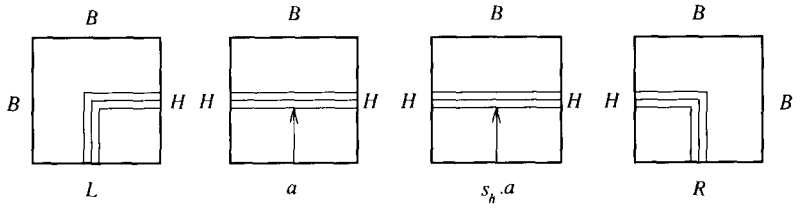

Fig. 5. The "halting" tiles.

With these tiles, we simulate the space–time diagram of a computation of the NDTM μ. It is clear that there exists a nontrivial tiling of the plane of length lower than $n \times n$ if and only if there exists a computation of μ that ends after less than $n-2$ steps. The number of colors of our set of tiles is exactly

$$6+(\sigma(\mu)+2)+2+2\cdot(\sigma(\mu)+2)=3\cdot\sigma(\mu)+14.$$

Our transformation is polynomial, hence FINITE-TILING is NP-complete. □

3.3. *From* FINITE-TILING *to* CA-FINITE-INJECTIVE

A transformation between tilings and two-dimensional cellular automata was first presented by Jarkko Kari in [6] and a more complete proof can be found in [7]. The main idea of the transformation is to introduce a special set of tiles which has an ad hoc property called *finite plane filling property*. We introduce another set of tiles,

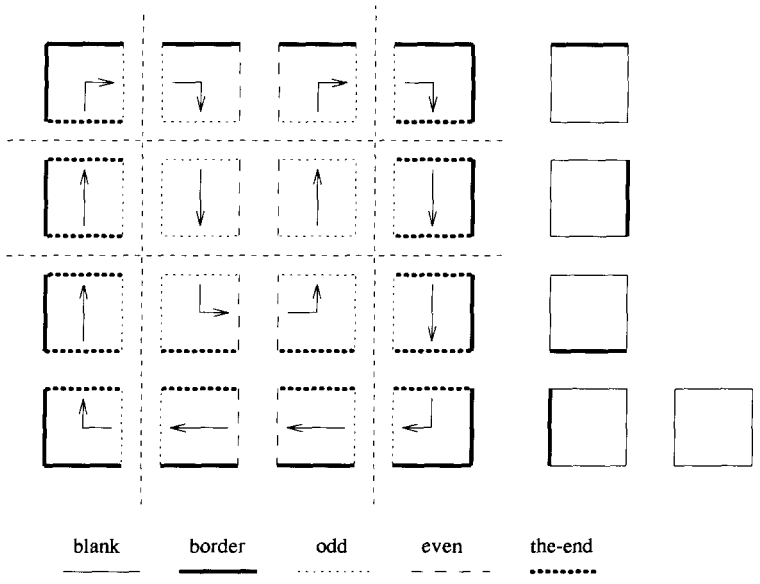

Fig. 6. The generic tiles of δ.

more simple than Kari's, which satisfies a slightly more restrictive property. We shall refer to our tile set as δ. With the help of δ, for each tile set τ, we construct a cellular automaton $\mathscr{A}_\tau$ in order to reduce FINITE-TILING to CA-FINITE-INJECTIVE.

CA-FINITE-INJECTIVE:
Instance: A two-dimensional cellular automaton $\mathscr{A}$ with Von Neumann neighborhood. Two integers p and q lower than the size of $\mathscr{A}$.
Question: Is $\mathscr{A}$ injective when restricted to all finite configurations smaller than $p \times q$?

Theorem 3. *CA-FINITE-INJECTIVE is co-*NP*-complete.*

3.4. The special set of tiles

Before proving this theorem, we introduce our tile set δ and its properties. The sides contain a color ("blank", "border", "odd", "even", or "the-end"), a label (N, S, E, W, N+, S+, E+, W+, or ω), and possibly an arrow. With this set of tiles, a tiling is considered as valid if and only if all pairs of adjacent sides have the same color, the same label, and for each arrow of the plane, its head points out on the tail of an arrow in the adjacent cell.

The tiles of δ can be found in Fig. 6. In this figure, the tiles are "generic" because labels are not represented. Each tile can have different labels which are defined below.

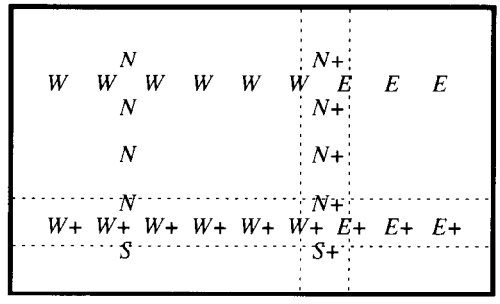

Fig. 7. The labels in a basic rectangle.

The labels are there to force that, in a valid tiling, inside a rectangle bordered with "border" tiles, there exists a unique cell labeled (N+, E+, S+, W+) (see Fig. 7 and Lemma 3). The labels of the other tiles should indicate whether the tile labeled (N+, E+, S+, W+) is above the cell, or on its right, etc.

The tiles of Fig. 6 with no arrow on them have their sides labeled ω. The four generic tiles with no side labeled "border" in the center-left of Fig. 6 have their North, East, South, and West sides labeled either (N+, E+, S+, W+), (X, Y, X, Y), (N, Y+, S, Y+) or (X+, E, X+, W) where X is N or S, and Y is E or W. In Fig. 6, the four upper generic tiles with arrows on them have their south side labeled N or N+, and the other sides labeled ω; the four left generic tiles have their west side labeled W or W+, and the other sides labeled ω; following the same scheme, E or E+ is on the right and S or S+ at the bottom.

We show in the rest of the section that our tile set δ has the desired properties.

Definition 1. A *basic rectangle* of size $p \times q$ is a finite valid tiling of the plane of size $p \times q$ with no side labeled "blank" or "border" inside the rectangle.

See Fig. 8 for a description of such a rectangle in which only the border cells and the arrows are represented.

Lemma 2. *Using the tile set δ, for all integers p and q, both greater than* 3, *there exists a basic rectangle of size $p \times 2q$. Each valid finite tiling of the plane consists of a finite number of juxtaposed basic rectangles.*

Proof. It is easy to convince oneself that a basic rectangle of size $p \times 2q$ can be obtained by using the row of tiles between the horizontal dashed lines and the two columns of tiles between the vertical dashed lines in Fig. 6. Note that the East side of a tile is "odd" if the number of tiles on its right until a border is odd. To prove that each valid finite tiling of the plane consists of a finite number of juxtaposed basic rectangles, we have to check if it is possible to find another kind of combinations of

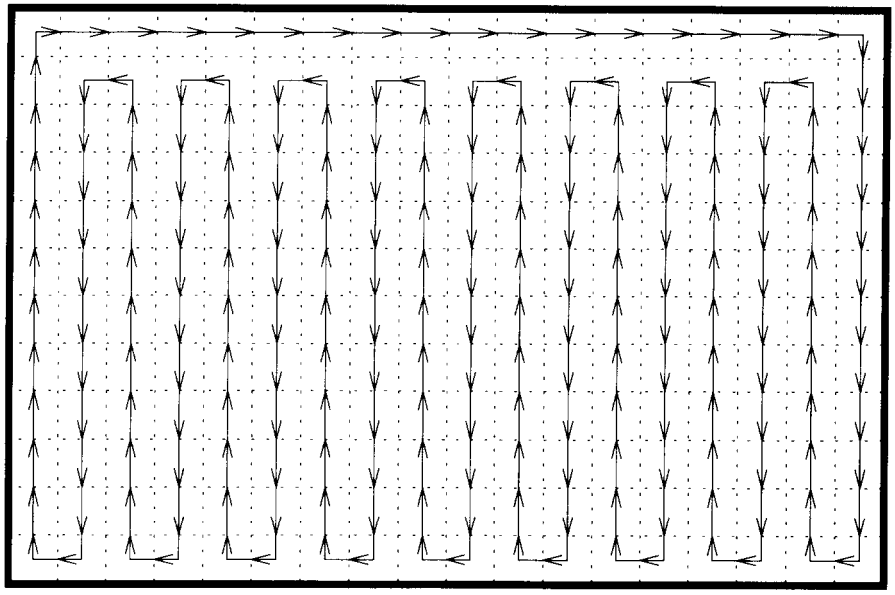

Fig. 8. A basic rectangle

these tiles. It is easy to convince oneself that if one considers a North-West nonblank tile, then it is a North-West corner of Fig. 6, and hence that it is followed on its right by a North border and below by a West border. By finite iteration, the result holds. □

Lemma 3. *Consider a basic rectangle. The path defined by the arrows of the cells forms a loop which visits one time each title of the inside of the rectangle. Inside the rectangle, there exists a unique cell labeled* (N+, E+, S+, W+).

Proof. Exactly as in the proof of Lemma 2, we observe that the leftmost column contains a vertical path directed North and that the loop in the rectangle is of the same kind as in Fig. 8. It is easy to check that there exists a unique cell labeled (N+, E+, S+, W+) in the rectangle: the cells with a side labeled $X+$ have to form a cross in the rectangle as shown in Fig. 7. The horizontal part of the cross is labeled horizontally W+ or E+, and above it the vertical labels are N, below they are S. □

By a proof similar to the proof of Lemma 2, we can show the following result.

Lemma 4. *Consider a finite tiling (valid or not). If the tiling is valid on each cell of a path, then this path forms a loop and visits every tile of a basic rectangle.*

3.4.1. The reduction

Consider a finite set C of colors with a blank color, and a collection $\tau \in C^4$ of tiles including a blank tile. We construct a cellular automaton $\mathscr{A}_\tau = (2, S, N, f_\tau)$ defined as follows:

- The *state* set S is included in $\delta \times \tau \times \{0.1\}$.
  S contains all triplets (d, t, α) of $\delta \times \tau \times \{0, 1\}$ under the following restrictions:
  - if one of the sides of d is "blank" or "border", then t is the blank tile of τ.
  - if d is labeled (N+, E+, S+, W+), then t *is not* the blank tile of τ.
- The *neighborhood* N is the Von Neumann neighborhood, i.e.

$$N = \{(0,0),(0,1),(0,-1),(1,0),(-1,0)\}.$$

- The *local rule* f_τ, applied on a cell the state of which is (d, t, α), may change only the bit component α. At each cell both the tilings δ and τ are checked. If there is a tiling error, or if the tile d contains no arrow, then the state of the cell is not altered. Otherwise, there is no tiling error in the concerned cell and the cell contains an arrow; the bit component is changed by performing an "exclusive or" operation with the bit attached to the cell pointed by the direction of the δ-component.
  The quiescent state of $\mathscr{A}_\tau$ is (blank, blank, 0).

We present now the basic theorem which provides a link between tilings and cellular automata.

Theorem 4. *Let n be an integer greater than or equal to 3 and τ be a set of tiles. The cellular automaton $\mathscr{A}_\tau$ is* not injective *restricted to finite configurations of size lower than $2n \times 2n$ if and only if the tile set τ* can be used *to form a finite nontrivial tiling of the plane of size lower than $(2n-4) \times (2n-4)$.*

Proof. Assume that $\mathscr{A}_\tau$ is not injective restricted to finite configurations. Then, there exist two different finite configurations c and c' having the same image by $\mathscr{A}_\tau$. Note that only the bits can be different in c and c' since $\mathscr{A}_\tau$ does not affect the tiles components: c and c' are different in at least one cell. On this cell, there is an arrow and the tilings are correct, otherwise the images of c and c' could not be same. Thus c and c' differ in the cell pointed by the arrow because an "exclusive or" is performed by $\mathscr{A}_\tau$. By finite induction, by Lemma 4, the constructed path forms a loop and there exists a basic rectangle of δ on which the tiling of τ is correct. By Lemma 3, the borders of the rectangle are blank in the state component of τ, hence we can construct a finite tiling with τ. The tiling is not trivial because τ is not blank on the cell labeled (N+, E+, S+, W+) in δ (Lemma 2). The size of the tiling is at most $(2n-4) \times (2n-4)$.

Conversely, assume that there exists a finite nontrivial tiling of the plane by τ of size lower than $(2n-4) \times (2n-4)$. We put this tiling inside a $2n \times 2n$ basic rectangle tiled by δ. The tiling is not trivial, thus there exists a nonblank tile on which we can put the tile of δ labeled (N+, E+, S+, W+). We define two configurations c and c' of size $2n \times 2n$: c is obtained by turning the bit component to 0 everywhere. For c', we keep the two tilings, and turn the bit component to 1 on the cells whose δ-component has an arrow,

to 0 elsewhere. As both tilings are correct, $\mathscr{A}_\tau$ performs an "exclusive or" on the loop of the rectangle and both c and c' have the same image (which is in fact c). Hence $\mathscr{A}_\tau$ restricted to finite configurations of size lower than $2n \times 2n$ is not injective. □

Proof of Theorem 3. With the previous result, it is very easy to prove that CA-FINITE-INJECTIVE is co-NP-complete. We prove a stronger result: CA-FINITE-INJECTIVE is co-NP-complete with a restriction on its instances; $p = q (= n)$ and $n \geqslant 2$ must be an even integer.

CA-FINITE-INJECTIVE with or without the restriction mentioned above is in co-NP because one can check in time polynomial in the size of the cellular automaton $\mathscr{A}$ if two finite configurations smaller than $n \times n$ have the same image by $\mathscr{A}$. So we only have to prove that the reduction presented from FINITE-TILING to CA-FINITE-INJECTIVE with the help of the automaton $\mathscr{A}_\tau$ has a polynomial cost. If c denotes the number of colors used by the tile set τ and t its number of tiles, then $t \leqslant c^4$. If we call d the size of the tile set δ then the size of the state set S of $\mathscr{A}_\tau$ is at most $2dt$. In the case of δ, the number of d its tiles is exactly 61. The number v of neighbors is 5, hence the size of $\mathscr{A}_\tau$ is $(2dt)^v \cdot \log(2dt)^v \, o((2dt)^v \cdot \log(2dt))$ which is bounded by a polynomial function of t. As each element of the transition table can be computed in constant time, the reduction between the two problems is polynomial. Notice that we have proved that under a polynomial reduction, the answer to the problem FINITE-TILING applied to the instance τ is "yes" if and only if the answer to the problem CA-FINITE-INJECTIVE applied to the instance $\mathscr{A}_\tau$ is "no". It explains why CA-FINITE-INJECTIVE is co-NP-complete and not NP-complete. □

4. Conclusion

In this paper, we have assumed that the size of the representation of a cellular automaton is the size of its transition table. But these tables are not always given extensively: a program which computes the value of the transition table on a given entry is often furnished. Thus it is natural to wonder if our complexity results remain true if we define the size of a CA as the length of the smallest program (or of the smallest Turing machine) which computes its transition table. Maybe reversible CA given by "simple" algorithms, when restricted to finite bounded configurations, have their inverses given by "simple" algorithms too.

References

[1] S. Amoroso and Y.N. Patt, Decision procedures for surjectivity and injectivity of parallel maps for tesselation structures, *J. Comput. System Sci.* **6** (1972) 448–464.

[2] R. Berger, The undecidability of the domino problem, *Mem. Amer. Math. Soc.* **66** (1966).

[3] S.A. Cook, The complexity of theorem-proving procedures, in: *Proc. 3rd Ann. ACM Symp. on Theory of Computing* (1971) 151–158.

[4] B. Durand, Undecidability of the surjectivity problem for 2D cellular automata: a simplified proof, in: *FCT'93*, Lecture Notes in Computer Science, (Springer, Berlin, 1993).
[5] B. Durand, The surjectivity problem for 2D cellular automata, *J. Comput. System Sci.*, to appear.
[6] J. Kari, Reversibility of 2D cellular automata is undecidable, *Physica* D **45** (1990) 379–385.
[7] J. Kari, Reversibility and surjectivity problems of cellular automata, *J. Comput. System Sci.*, to appear.
[8] E.F. Moore, Machine models of self-reproduction, *Proc. Symp. Appl. Math.* **14** (1962) 13–33.
[9] J. Myhill, The converse to Moore's garden-of-Eden theorem, *Proc. Amer. Math. Soc.* **14** (1963) 685–686.
[10] D. Richardson, Tesselations with local transformations, *J. Comput. System Sci.* **6** (1972) 373–388.
[11] R.M. Robinson, Undecidability and nonperiodicity for tilings of the plane, *Invent. Math.* **12** (1971) 177–209.
[12] K. Sutner, De Bruijn graphs and linear cellular automata, *Complex Systems* **5** (1991) 19–30.
[13] J.D. Ullman, A.V. Aho and J.E. Hopcroft, *The Design and Analysis of Computer Algorithms* (Addison-Wesley, Reading, MA, 1974).
[14] J.D. Ullman and J.E. Hopcroft, *Introduction to Automata Theory, Languages and Computation* (Addison-Wesley, Reading, MA, 1979)