Physica D 45 (1990) 379–385
North-Holland

REVERSIBILITY OF 2D CELLULAR AUTOMATA IS UNDECIDABLE

Jarkko KARI
Mathematics Department, University of Turku, 20500 Turku, Finland

Received 8 January 1990
Revised manuscript received 20 February 1990

The problem whether a given two- or higher-dimensional cellular automaton is reversible is shown to be algorithmically undecidable. A cellular automaton is called reversible if it has an inverse automaton, that is a cellular automaton that makes the system retrace its steps backwards in time. The same problem is known to be decidable for one-dimensional automata.

1. Introduction

Cellular automata are discrete and deterministic dynamical systems. They are especially suitable for modeling natural systems that can be described as massive collections of simple objects interacting locally with each other. A d-dimensional cellular automaton consists of an infinite d-dimensional array of identical cells. Each cell is always in one state from a finite state set. The cells alter their states synchronously on discrete time steps according to a local rule. The rule gives the new state of each cell as a function of the old states of some nearby cells, its neighbors. The array is homogeneous so that all its cells operate under the same local rule. The states of all the cells in the array are described by a configuration. A configuration can be considered as the state of the whole array. The local rule of the automaton specifies a global function that tells how each configuration is changed in one time step.

Cellular automata provide simple models of complex natural systems encountered in physics, biology and other fields. Like natural systems they consist of large numbers of simple basic components that together produce the complex behavior of the system. A basic feature of microscopic mechanisms of physics seems to be reversibility [10]. It is possible for cellular automata to capture this important characteristic without sacrificing other essential properties like computational universality. A cellular automaton rule is called reversible if there exists another rule, called the inverse rule, that makes the automaton retrace its computation steps backwards in time. The earlier configurations are uniquely determined by the present one and no information is lost during the computation. A cellular automaton defined by a reversible local rule is called reversible. It is known that a cellular automaton is reversible if its global function is one-to-one [8].

It is a natural question to ask what kind of local rules are reversible. No general characterizations are known. In fact, there does not exist an algorithm that would decide whether a given local rule is reversible or not. This fact is the main result reported in this article. If only one-dimensional rules are considered, then such an algorithm can be designed [1].

Another property a cellular automaton can have is surjectivity, that is the surjectivity of its global function. In a surjective cellular automaton every configuration can occur arbitrarily late during computations. It is known that an automaton is surjective if and only if the restriction of its global function to finite configurations is injective. This is the so-called Garden of Eden theorem proved by Moore [6] and Myhill [7]. A configuration is called finite if it has only finitely many cells in states different from one specific quiescent state. One can

show, using a similar method as in connection with the reversibility, the algorithmic undecidability of the problem of testing whether a given cellular automaton is surjective. In the one-dimensional case the problem is, however, decidable [1].

The article is organized as follows. First the concepts used in the paper are formally defined. Then we outline the proof for the undecidability of the reversibility problem. The idea of the proof is very simple, but the complete proof contains a number of confusing details and it is unnecessary to go into these details in the present article. The full proof can be found in refs. [4,5]. In section 4 surjective cellular automata are considered. We briefly discuss the differences between the reversibility and surjectivity problems, and we present how the ideas of section 3 can be used to show the surjectivity problem undecidable. The final section contains some concluding remarks and consequences of the results.

2. Definitions

Let us first explain the notions used. Formally, a *cellular automaton* (CA) is a quadruple

$$\mathcal{A} = (d, \mathrm{S}, N, f),$$

where d is a positive integer indicating the *dimension* of $\mathcal{A}$, S is a finite *state set*, N is a *neighborhood vector*

$$N = (\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n)$$

of n different elements of Z^d and f is the *local rule* of the CA presented as a function from S^n into S. The cells are laid on an infinite d-dimensional array and their positions are indexed by Z^d, the set of d-tuples of integers. The *neighbors* of a cell situated in $\bar{x} \in Z^d$ are the cells in positions

$$\bar{x} + \bar{x}_i, \qquad \text{for } i = 1, 2, \ldots, n.$$

The local rule f gives the new state of a cell from the old states of its neighbors.

A *configuration* of a CA $\mathcal{A} = (d, \mathrm{S}, N, f)$ is a function

$$c : \mathrm{Z}^d \to \mathrm{S}$$

that assigns states to all cells. Let C denote the set of all configurations. The local rule f determines the *global function*

$$G_f : C \to C$$

that describes the dynamics of the CA. At each time step a configuration c is transformed into a new configuration $G_f(c)$ where

$$G_f(c)(\bar{x}) = f(c(\bar{x} + \bar{x}_1), c(\bar{x} + \bar{x}_2), \ldots, c(\bar{x} + \bar{x}_n))$$

for all $\bar{x}$ in Z^d. For each state $s \in \mathrm{S}$ let $conf(s)$ denote the homogeneous configuration where all the cells are in the same state s.

Sometimes a *quiescent state* q in S is distinguished. The quiescent state must have the property

$$f(q, q, \ldots, q) = q.$$

The configuration $conf(q)$ with all cells in the quiescent state is called the *quiescent configuration*. A configuration is called *finite* if it has only finitely many cells in non-quiescent states. Let C_{F} denote the set of finite configurations. It follows from the special property of the quiescent state that a finite configuration remains finite in the evolution of the CA. Let G_f^{F} denote the restriction of the global function G_f to the set of finite configurations.

In this work mainly two-dimensional cellular automata are considered. In this case the cells are laid on the plane. The following two-dimensional neighborhood vector will be frequently used :

$$N_{\mathrm{vN}} = [(0,0), (1,0), (0,1), (-1,0), (0,-1)].$$

This defines the so called *von Neumann neighborhood* – each cell has five neighbors: the cell itself and the four adjacent cells (see fig. 1). The four directions of the compass are used when we refer to the four adjacent neighbors.

A CA is called *injective* if its global function is one-to-one. Similarly, a CA is called *surjective* if its global function is surjective.

Example Let

$$N = (\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n)$$

be any neighborhood vector of a d-dimensional CA,

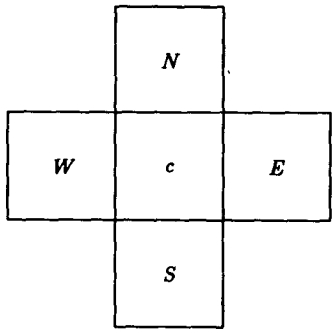

Fig. 1. The von Neumann neighborhood of cell c.

$S = S_1 \times S_2 \times \ldots \times S_n$

a Cartesian product of n finite sets and

φ: $S \to S$

a bijective function. Let

π_i: $S_1 \times S_2 \times \ldots \times S_n \to S_i$

denote the ith projection of S. Let f be a function from S^n into S defined by

$$f(s_1, s_2, \ldots, s_n) = \varphi(\pi_1(s_1), \pi_2(s_2), \ldots, \pi_n(s_n)) .$$

It is easy to see that the d-dimensional cellular automaton $\mathcal{A} = (d, S, N, f)$ is injective. This follows from the bijectivity of φ – if you change its arguments then its value is changed as well. Define then another function g from S^n into S by

$$g(s_1, s_2, \ldots, s_n)$$

$$= (\pi_1(\varphi^{-1}(s_1)), \pi_2(\varphi^{-1}(s_2)), \ldots, \pi_n(\varphi^{-1}(s_n)))$$

and let N^{-1} be the neighborhood obtained from N by changing the signs of all of its coordinates. Then the CA $\mathcal{B} = (d, S, N^{-1}, g)$ is injective and, moreover, it is the inverse automaton of $\mathcal{A}$. This means that $G_g = G_f^{-1}$. □

The injective CA in the example above were *reversible*, that is, they had an *inverse automaton*. Richardson [8] showed that this is the case with every injective CA.

3. The reversibility problem

The reversibility problem asks whether a given CA rule is reversible, or equivalently whether the global function it defines is injective. In this section we present the basic ideas how one can prove the reversibility problem undecidable in the case of two-dimensional CA. The complete proof is given in refs. [4,5].

The proof is essentially a reduction from a well known undecidable problem, the so-called *tiling problem*. In the tiling problem we are given a finite set of unit squares with colored edges, the tiles, placed with their edges horizontal and vertical. We have infinitely many copies of all these tiles and we want to tile the entire plane using the copies, without rotating any of them. The tiles are placed to the unit square regions of the Euclidean plane bordered by the lines $y = n$ and $x = m$ for all integers n and m. In a valid tiling the abutting edges of the tiles must have the same color. The tiling problem consists of deciding whether the plane can be tiled with a given collection of tiles. The tiling problem was proved undecidable by Berger [2]. A simplified proof was given later by Robinson [9].

An easy application of König's infinity lemma shows that if one can tile arbitrarily large squares with a given tile set, then one can tile the whole plane as well. This fact will be used in our proof.

In the proof a specific set $\mathcal{D}$ of *directed tiles* is needed. The directed tiles are, like ordinary tiles, unit squares with colored edges. In addition, there is one direction from the set

$$\{N, E, S, W\}$$

attached to every tile. The direction refers to one of the four neighboring tiles lying to the north, east, south or west.

The directions define paths through the tiles on the plane in a natural way. The direction of each tile tells which way a path coming to the tile proceeds. The next tile on the path is the adjacent tile pointed by the direction. Every tile on the plane starts a path that follows the directions. The path is not necessarily infinite – it may come back to an earlier tile causing the path to fall into a loop.

Next we define a special property that the set $\mathcal{D}$ of directed tiles is supposed to satisfy. We call it the *plane filling property*. First, it is required that there exists a valid tiling of the plane with the tiles. Secondly, the essential requirement is that on every tiling of the plane with the tiles, valid

or not, the directions can define only two different types of paths: Either there is a tile on the path where the tiling is incorrect, or otherwise the path visits all tiles of arbitrary large squares. So if the tiling property is not violated on the path then for each n there must be an $n \times n$ square each tile of which is on the path. Especially the directions cannot define any loop without a violation of the tiling property somewhere along the path.

The second requirement can be illustrated as follows. Consider a simple memoryless device that operates on the plane which is tiled with the directed tiles of $\mathcal{D}$. The device checks the tiling at the tile it is currently standing on. This means it looks at the four tiles that have an abutting edge with its own tile, and it checks whether the four pairs of abutting edges have the same color. If the tiling is proper, then the device goes to the neighboring tile that is pointed by the direction of its current tile. The operation is repeated on the new tile. If, on the other hand, the tiling property is violated at the tile, then the device halts, indicating that it has found an error.

If the tile set $\mathcal{D}$ has the plane filling property then, no matter how the plane is tiled with the tiles, and no matter which tile the device is started at, there are just two possible ways the device can operate. Either it halts because it finds a tiling error, or it visits all tiles of arbitrarily large squares.

The following proposition allows us to use the tile set $\mathcal{D}$ in our undecidability proof. The proof of the proposition is omitted. It can be found in refs. [4,5]. The proof consists of an explicit construction of the tile set with the plane-filling property. The plane-filling property is defined in a slightly different way in refs. [4,5], but the tile set can be modified rather easily to satisfy the formulation we have used here.

Proposition 1 *There exists a set $\mathcal{D}$ of directed tiles that satisfies the plane filling property.*

In fig. 2 there is a portion of the path defined by the directed tiles of $\mathcal{D}$. The path has the shape of the well known Peano curve.

Once we have the tile set $\mathcal{D}$ the reduction of the tiling problem to the injectivity problem of two-dimensional CA is very simple. All the complicated details of the reduction are contained in the construction of the tile set with the plane filling property.

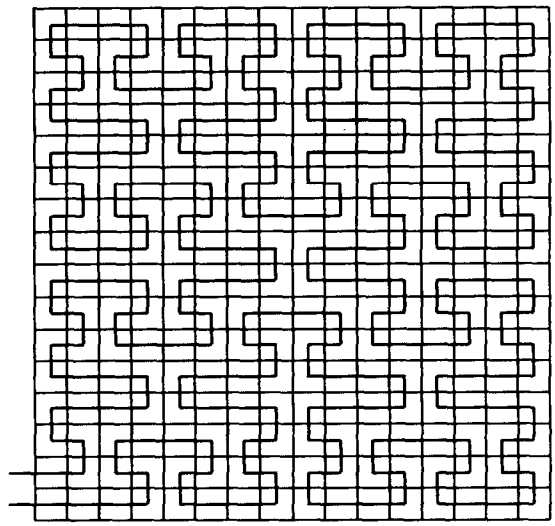

Fig. 2. A part of the plane-filling curve defined by the directed tiles.

Let $\mathcal{T}$ be any ordinary tile set, that is, a set of tiles without directions. One can construct a two-dimensional cellular automaton

$$\mathcal{A}_{\mathcal{T}} = (2, \mathcal{D} \times \mathcal{T} \times \{0,1\}, N_{\mathrm{vN}}, f_{\mathcal{T}})$$

such that $\mathcal{A}_{\mathcal{T}}$ is not injective if and only if the tile set $\mathcal{T}$ can be used to tile the plane.

The automaton $\mathcal{A}_{\mathcal{T}}$ uses the von Neumann neighborhood. Its states contain two tile components, one from $\mathcal{D}$ and one from $\mathcal{T}$. In addition there is a bit, 0 or 1, attached to each state. The local rule $f_{\mathcal{T}}$ may change only the bits. The tile components are never changed. At each cell the tilings with both the $\mathcal{D}$- and $\mathcal{T}$-components are checked. If there is a tiling error in either of the components then the state of the cell is not altered. If the tilings are correct then the bit component is changed by performing the exclusive or (= XOR) operation with the bit that is attached to the cell pointed by the $\mathcal{D}$-component. The XOR operation is the same as addition of the bits modulo 2.

Suppose that there exists a tiling of the plane with the tile set $\mathcal{T}$. Construct two configurations c_0 and c_1 of $\mathcal{A}_{\mathcal{T}}$ as follows. The tile components of c_0 and c_1 constitute the same legal tilings of the plane with the tiles of $\mathcal{D}$ and $\mathcal{T}$. In c_0 all bits are

0 while in c_1 they equal 1. Because the tilings are correct everywhere each bit is changed using the XOR operation with the next bit on the path. This means that in both configurations c_0 and c_1 the bits are all changed to 0. So in $\mathcal{A}_\mathcal{T}$ there are two different configurations c_0 and c_1 that are turned into the same configuration in one time step and $\mathcal{A}_\mathcal{T}$ cannot be injective.

Conversely, suppose that $\mathcal{A}_\mathcal{T}$ is not injective. Let c_0 and c_1 be two different configurations that $\mathcal{A}_\mathcal{T}$ turns into the same configuration in one time step. The tile components of c_0 and c_1 must coincide. Consider a cell whose bit component is different in c_0 and c_1. The tilings must be correct at the cell, and the bits are different also in the cell pointed by the $\mathcal{D}$-component. The reasoning can be repeated for this next cell. By continuing through succeeding cells it is concluded that the tiling properties may not be violated at any of them. Since $\mathcal{D}$ has the plane filling property this path goes through arbitrary large squares. So arbitrary large squares can be tiled using the tiles of $\mathcal{T}$. This means that the whole plane can be tiled.

We showed that there is a valid tiling of the plane using the tile set $\mathcal{T}$ if and only if the cellular automaton $\mathcal{A}_\mathcal{T}$ is not reversible. If there were an algorithm for solving the reversibility problem, then this algorithm applied to $\mathcal{A}_\mathcal{T}$ would solve the tiling problem. This is not possible because the tiling problem is undecidable.

Theorem 1 *It is undecidable whether a given two-dimensional cellular automaton with the von Neumann neighborhood is reversible.*

4. The surjectivity problem

A cellular automaton is called surjective if its global transition function is surjective. Surjectivity is a less restricted property of cellular automata than injectivity, since every injective rule is also surjective. This follows immediately from the well known Garden of Eden theorem of Moore [6] and Myhill [7], which states that the global function G_f is surjective if and only if its restriction G_f^F to finite configurations is injective. This means that instead of testing the surjectivity property one can test the injectivity of G_f^F.

The two injectivity problems – the problem of testing the injectivity of G_f on one hand, and the problem of testing the injectivity of G_f^F on the other hand – have one basic difference. The cellular automata with injective G_f can be *effectively enumerated* by listing all CA and checking for each pair if they are inverses of each other. If they are, they are added to the list of injective CA. Remember that a CA is injective if and only if it is reversible. On the other hand, we cannot enumerate the non-injective CA, because the injectivity problem is undecidable.

With the restriction G_f^F the situation is just the opposite: One can enumerate the CA having a non-injective G_f^F but, as theorem 2 below states, one cannot enumerate the CA with injective G_f^F. In the previous section the reduction from the tiling problem was done in such a way that the tiling problem had a solution iff the corresponding G_f was not injective. This reduction can be done since the tile sets that possess a solution cannot be effectively enumerated. Doing the same with G_f^F is not possible. A similar reduction would map the tile sets with valid tilings into the effectively enumerable set of non-injective G_f^F.

This problem is solved by introducing a new tiling problem, called *the finite tiling problem.* The tile sets that possess the finite tiling property can be effectively enumerated. This is why this problem seems better suited for our purpose.

In the finite tiling problem we are given a finite collection of tiles. The tiles have colored edges as before. One of the tiles is called *blank.* All of its edges have the same color. In a valid tiling of the plane the abutting edges of adjacent tiles must have the same color, just like before. There is of course at least one valid tiling – namely the one where all the tiles are blank. This tiling is called *trivial.* A tiling is called *finite* if only finitely many of the tiles on the plane are not blank. The problem is to decide whether there is a valid, finite and non-trivial tiling of the plane. This problem can easily be proved undecidable [4].

Proposition 2 *It is undecidable whether a given collection of tiles containing a blank tile can be used to form a finite tiling that is valid but not trivial.*

Reducing the finite tiling problem to the injec-

tivity problem of G_f^F can be done using a similar method as in section 3. The set $\mathcal{D}$ of directed tiles must, however, be changed a little to allow finite tilings of the plane. Let $\mathcal{D}_0$ be the altered tile set (see ref. [4] for its construction). It contains one blank tile that does not have a direction, and instead of the plane-filling property the set satisfies the following property. For every non-trivial tiling of the plane and for every path defined by the directions of the tiles the following is true: If the tiling property is not violated on the path then either the path visits all tiles of arbitrarily large squares (as in section 3), or the path is a closed loop visiting all tiles of some square surrounded by blank tiles. In the latter case a specific tile d_0 is known to be at the center of the square, but not anywhere else inside the square. It is also known that for arbitrary large squares there are tilings with a closed path of the latter type through the square.

Let $\mathcal{T}$ be any tile set containing a blank tile. Let us construct a cellular automaton $\mathcal{B}_\mathcal{T}$ whose states have two tile components, one from $\mathcal{T}$ and one from $\mathcal{D}_0$. The components satisfy the following two restrictions:

(i) If the $\mathcal{D}_0$-component is blank then the $\mathcal{T}$-component is also blank, and

(ii) if the $\mathcal{D}_0$-component equals d_0 then the $\mathcal{T}$-component is *not* blank.

The state whose both components are blank is the quiescent state. Every non-quiescent state contains one bit.

The local rule of $\mathcal{B}_\mathcal{T}$ is defined in the same way as in section 3. At each cell the tilings with both $\mathcal{T}$- and $\mathcal{D}_0$-components are checked. If the tiling property is violated in either of the components, then the state of the cell is not changed. If, on the other hand, the tilings are valid at the cell, then the bit is changed by performing an exclusive or operation with the bit attached to the cell pointed by the direction of the $\mathcal{D}_0$-component. The quiescent state is never altered.

It is an easy matter to show that the restriction of the global function of $\mathcal{B}_0$ to finite configurations is injective if and only if the tile set $\mathcal{T}$ cannot be used to form a valid, non-trivial and finite tiling of the plane. Note how finite tilings correspond nicely to finite configurations. Since the finite tiling property is undecidable, and since a CA is surjective iff the restriction of its global function is injective, we have proved:

Theorem 2 *It is undecidable whether a given two-dimensional cellular automaton with the von Neumann neighborhood is surjective.*

5. Conclusion

Our main theorem, which states the undecidability of the reversibility problem, has interesting consequences. By definition, every reversible CA has an inverse automaton. But even though the original automaton uses the von Neumann neighborhood, its inverse may need a much wider neighborhood. In fact, no computable function of the size of the state set can be an upper bound for the inverse neighborhood. If there were a computable bound, then one could generate all candidates for the inverse CA and check one after the other whether some of them really is the inverse automaton. Note that it is an easy matter to check whether two given CA are the inverses of each other. This would yield an algorithm for the injectivity problem, which is not possible.

Theorem 1 implies that given a reversible two-dimensional CA rule finding its inverse is very difficult in general. There is no algorithm for finding the inverse rule with a time complexity bounded by a computable function.

This fact may have direct applications in public-key cryptography. Reversible CA rules can be used to encrypt messages in a natural way: The plain text is expressed as a configuration of the CA, and the encryption is done by applying the CA rule for a fixed number of time steps. The configuration obtained is the cryptotext. One may use periodic boundary conditions to make the configurations finite in size. The decryption is done simply by applying the inverse rule to the cryptotext for the same number of steps. The operations can be done very efficiently in parallel if proper hardware implementations are available.

If one is using an arbitrary two-dimensional reversible cellular automaton $\mathcal{A}$ to encrypt and its inverse automaton $\mathcal{A}^{-1}$ to decrypt messages, then one can make $\mathcal{A}$ public. Making $\mathcal{A}$ public does not reveal its inverse $\mathcal{A}^{-1}$.

One should be aware that theorem 1 does not deny the possibility that there exist simple characterizations of reversible CA rules. It would be very useful to have some general representation of reversible rules. However, such a representation must naturally be undecidable, so that we cannot test whether a given rule can be presented in the suggested form.

Acknowledgements

I am indebted to Arto Salomaa and Anatoli Bolotov for their comments and suggestions.

References

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

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

[3] K. Culik II, On invertible cellular automata, Complex Systems 1 (1987) 1035–1044.

[4] J. Kari, Decision problems concerning cellular automata, Ph.D. Thesis, University of Turku, Finland (1989).

[5] J. Kari, Reversibility and surjectivity problems of cellular automata, Computer System Sci., submitted for publication.

[6] E.F. Moore, Machine models of self-reproduction, Proc. Symp. Appl. Math. 14 (1962) 17–33.

[7] J. Myhill, The converse to Moore's Garden-of-Eden theorem, Proc. Am. Math. Soc. 14 (1963) 685–686.

[8] D. Richardson, Tessellations with local transformations, J. Computer System Sci. 6 (1972) 373–388.

[9] R.M. Robinson, Undecidability and nonperiodicity for tilings of the plane, Inventiones Mathematicae 12 (1971) 177–209.

[10] T. Toffoli and N. Margolus, Cellular Automata Machines (MIT Press, Cambridge, MA, 1987).

[11] S. Wolfram, Theory and Applications of Cellular Automata (World Scientific, Singapore, 1986).