Complex Systems **2** (1988) 177–190

Undecidability of CA Classification Schemes*

Karel Culik II
Department of Computer Science, University of South Carolina, Columbia, SC 29208, USA

Sheng Yu
Department of Mathematical Sciences, Kent State University, Kent, OH 44242, USA

Abstract. Stephen Wolfram introduced the use of cellular automata as models of complex systems and proposed a classification of these automata based on their statistically observed behavior. We investigate various properties of these classes; in particular, we ask whether certain properties are effective, and we obtain several somewhat surprising results. For example, we show that it is undecidable whether all the finite configurations of a given cellular automaton eventually become quiescent. Consequently, it is undecidable to which class a given cellular automaton belongs, even when choosing only between the two simplest classes.

1. Introduction

In recent years, cellular automata (CA) have been extensively investigated as mathematical models of complex systems in physics, biology, etc. In [7], Wolfram suggested a classification of cellular automata by the qualitative features of cellular automaton evolution. He observed that cellular automata appear to fall into the following four classes:

1. Evolution leads to a homogeneous state.
2. Evolution leads to a set of separated simple stable or periodic structures.
3. Evolution leads to a chaotic pattern.
4. Evolution leads to complex localized structures, sometimes long-lived.

*This research was supported by the National Science Foundation under Grant No. CCR-8702752.

In this paper, we propose a more specific definition of these four classes of CA. We will classify CA by their behavior on *finite configurations* even if the computations on all possible configurations might be of interest. A configuration is finite if all but finite number of cells are in the same state (usually called the quiescent state). It is convenient for the presentation of our results to define the classes as a hierarchy, i.e. a higher class includes each lower class as a subset. It is, of course, easy to modify this definition, i.e. to define Class Two as our Class Two minus Class One. Informally, the four classes are described as follows. In Class One, all the finite configurations evolve into a quiescent configuration. In Class Two, all the finite configurations have an ultimately periodic evolution. In Class Three there are CAs for which it is decidable whether α ever evolves to β for two given configurations α and β. Class Four includes all CAs. The formal definitions of our classes are in sections where they are studied, i.e. Class One in section 3, Class Two in section 4, and Class Three and Four in section 5. Note that our classification is not restricted to the totalistic CAs. When we compare our classification with that of Wolfram we really mean the union of Wolfram classes up to class k where we talk about Wolfram's class k. The first class as defined here appears to be in general somewhat larger when compared to Wolfram's. However, our definition coincides with Wolfram's classification of totalistic cellular automata with number of states $k = 2$, and neighborhood size $r = 2$, which were shown as examples in [7]. The precise definitions allow us to prove certain properties of these classes, rather than just speculate about them.

In the next section, we give basic definitions of cellular automata. In section 3, 4, and 5 we define and study the properties of Class One, Class Two, and Class Three and Four, respectively. In section 6 we introduce the products of cellular automata.

We investigate various properties of the classes, and we ask whether certain properties are effective. We show that it is undecidable whether all the finite configurations of a given cellular automaton eventually become quiescent, and show that the fact that all finite configurations evolve to the quiescent configuration does not imply that the limit language is regular. We also show that all configurations of a cellular automata evolve to the quiescent configuration if and only if all the *finite* configurations evolve to the quiescent configuration in a bounded (i.e. constant) time. Using our definition, we can easily show that universal cellular automata [1] are not In Class Three, and that the product of two cellular automata is always in the higher class of the two. On the other hand, it is undecidable to which class a cellular automaton belongs, even when choosing only between Class One and Class Two.

We hope this discussion will help bring some insight into the basic properties of cellular automata.

2. Basic definitions

A cellular automaton is a quadruple $A = (k, S, N, f)$, where $k \geq 1$ is the dimension, S is the finite set of states, N is the *neighborhood*, and f is the *local function*. Here, we mainly consider linear (one-dimensional) cellular automata. Therefore, the dimension k will be omitted in the following. However, all the results except the one on limit languages can be easily extended to multi-dimensional cellular automata. The *neighborhood* N is a sequence $\{I_1, I_2, \ldots, I_h\}$ of relative locations $I_j \in Z$, $1 \leq j \leq h$. Consequently, the neighborhood of a cell I is the sequence of cells

$$N(I) = \{I + I_1, I + I_2, \ldots, I + I_h\}.$$

The local function $f : S^h \to S$ is a total function which computes the next state of a cell I from the current states of all cells in its neighborhood $N(I)$.

A *configuration* c is a function $c : Z \to S$, which assigns a state in S to each cell of the CA. The set of all configurations is denoted S^Z. The local function f is extended to *global function* $G_f : S^Z \to S^Z$ such that, for any configurations c_1, $c_2 \in S^Z$,

$$G_f(c_1) = c_2$$

if and only if

$$c_2(I) = f(c_1(I + I_1), c_1(I + I_2), \ldots, c_1(I + I_h)),$$

for all I in Z.

A state s of a cellular automaton A is called a *stable state* if the following condition holds:

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

A configuration with all cells being in a stable state s is called a *homogeneous configuration* of s. A configuration with all but finitely many cells in the state s is called an *s-finite configuration.*

A stable state q of A is distinguished and called the *quiescent state* of A. The homogeneous configuration of q is called the *quiescent configuration.* A q-finite configuration is simply called a *finite configuration.*

A *segment* is a finite, consecutive part of a configuration surrounded by quiescent cells and such that no cell in it is quiescent.

Let c_1 and c_2 be two configurations of A. If $G_f{}^i(c_1) = c_2$ for some $i \geq 0$, then we say that c_1 evolves to c_2. For a configuration c, the infinite sequence

$$c,\ G_f(c),\ G_f{}^2(c),\ \ldots,\ G_f{}^i(c),\ \ldots$$

is called the *evolution sequence* of c. If the sequence is ultimately periodic, then we say that c has an ultimately periodic evolution. The set

$$D(c) = \{\, d \mid d = G_f{}^i(c) \text{ for some } i \geq 0 \,\}$$

is called the *descendant set* of c. We are only interested in the finite non-s part of an s-finite configuration. Therefore, for each stable state s we define a mapping $\pi_s : S^Z \to S^*$, which maps each configuration c into a word w the minimum sequence of states in c that covers all the non-s state in c. If c is an s-finite configuration then $\pi_s(D(c))$ is called the *descendant language* of c with respect to s.

In [7], totalistic cellular automata are extensively used as examples of the four classification classes. In this paper, we use the same notation for those totalistic cellular automata. We use k to denote the number of states, i.e. $S = \{0, 1, \ldots, k-1\}$, and r to denote the neighborhood radius, i.e. $N = \{-r, \ldots, 0, \ldots, r\}$. A function code of a totalistic cellular automaton with $k = 2$ is denoted by a binary number

$$b_{2r+1} \ldots b_2 b_1 b_0,$$

for $b_i \in \{0, 1\}$, $0 \leq i \leq 2r+1$, and it means that

$$f(s_{-r}, \ldots, s_0, \ldots s_r) = b_i, \text{ if } s_{-r} + \ldots + s_0 + s_r = i.$$

Usually, we convert these binary function codes into decimal numbers for the purpose of convenience.

3. Class One

We propose the following definition for Class One cellular automata.

Definition 1. *A cellular automaton $A = (S, N, f)$ is in Class One if there is a stable state $s \in S$ such that all the s-finite configurations of A, evolve to the homogeneous configuration of s.*

Example 1. *The cellular automaton with $k = 2$, $r = 2$ and function code 4 is in Class One. It can be shown that every finite configuration of this CA evolves to a shorter one in at most five steps.*

The reader can verify that the cellular automata with $k = 2$, $r = 2$ and function codes 0, 4, 16, 32, 36, 48, 54, 60, and 62, are all in Class One by our definition. Moreover, no other cellular automaton with $k = 2$ and $r = 2$ is in Class One. Hence, our definition of Class One agrees with Wolfram's definition for totalistic CA with $k = 2$ and $r = 2$. However, our definition for Class One appears to be a larger class for more complex cellular automata.

Note that a Class One cellular automaton, by either Wolfram's or our definition, can have "exceptional" configurations that do not evolve to a homogeneous configuration. See the following example.

Example 2. *The following (infinite) configuration of the Class One cellular automaton with $k = 2, r = 2$ and function code 16 never evolves to a homogeneous configuration:*

$$\ldots 0\,1\,0\,1\,0\,1\,0\,1\,0\,1 \ldots$$

Theorem 1. *For each cellular automaton, the following four statements are equivalent:*

1. *All configurations evolve to a homogeneous configuration.*
2. *All configurations evolve to a homogeneous configuration in a bounded (constant) time.*
3. *All s-finite configurations evolve to the homogeneous configuration of s in a bounded (constant) time.*
4. *The limit set consists of the homogeneous configuration of s only.*

Proof. The equivalence of the first two statements can be easily proved by considering the stable state as the quiescent state and applying corollary 2 of [2]. Statements 2 and 3 are clearly equivalent. Statement 2 implies statement 4. Statement 4 implies statement 1. ∎

Example 2 shows that if we defined Class One to be the set of all cellular automata in which all the configurations evolve to a homogeneous configuration, then some of Wolfram's simple examples, e.g. cellular automata with function code 4, 16, 32, 36, ... and $k = 2$, $r = 2$, would not be included. By theorem 1 and the definition of Class One none of the above statements is implied by the fact that a cellular automaton is in Class One.

Theorem 1 says that in general there does not exist a constant bound for a Class One cellular automaton such that every finite configuration evolves to a homogeneous configuration within this bound. But is there a linear bound or a polynomial bound in terms of the lengths of (the non-s parts of) s-finite configurations? We first look at an example.

Example 3. *A cellular automaton $A = (S, N, f)$ is defined as follows: $S = \{0, 1, -, q\}$, $N = (-1, 0, 1)$, and f:*

$$\begin{array}{lll} f(q,0,x) = q; & f(q,-,x) = q; & f(x,1,q) = 0; \\ f(y,0,q) = -; & f(y,-,q) = 0; & f(y,-,z) = 1; \\ f(y,-,-) = 0; & f(y,0,-) = -; & f(x,1,-) = 0; \end{array}$$

for $x = 0$, 1, $-$, or q; $y = 0$, 1, or $-$; $z = 0$ or 1; and

$$f(a,b,c) = b, \text{ for all other cases.}$$

A table for f without shorthand notations is available in Appendix B for the readers who are interested in testing this CA. Following is an sample evolution of A starting with a segment 1011 (surrounded by q's):

```
(1) ... q 1 0 1 1 q ...     (8)   ... q 0 1 0 0 q ...
(2) ... q 1 0 1 0 q ...     (9)   ..... q 1 0 - q ...
(3) ... q 1 0 1 - q ...    (10)   ..... q 1 - 0 q ...
(4) ... q 1 0 0 0 q ...    (11)   ..... q 0 1 - q ...
(5) ... q 1 0 0 - q ...    (12)   ....... q 0 0 q ...
(6) ... q 1 0 - 0 q ...    (13)   ......... q - q ...
(7) ... q 1 - 1 - q ...    (14)   ........... q .....
```

The reader can verify that a segment is treated as a variation of a binary number and it is subtracted by one at each evolution step.

In the above example, a segment may evolve for an exponential number of steps in terms of its length until it becomes all quiescent. In fact, for any computable function $T(n)$, there exists a Class One cellular automaton such that some of its s-finite configurations need $T(n)$ steps to evolve to the quiescent configuration (homogeneous configuration of s).

Theorem 2. *There exists a one-dimensional Class One cellular automaton whose limit language is not regular.*

Proof. Let $A = (S, N, f)$ be a one-dimensional cellular automaton that has a nonregular limit language L. See [4] for such a cellular automaton. We construct a cellular automaton $A' = (S', N, f')$, where $S' = S \cup \{q\}$, $q \notin S$ and q is the quiescent state of A'. The local function f' is the same as f on all the states in S, and maps any neighborhood that contains state q. Obviously, all the finite configurations of A' evolve to the quiescent configuration. Therefore, A' is in Class One. Let L' be the limit language of A'. Note that $L' \cap S^* = L$ and L is not regular. By the closure property of regular languages, L' is not regular. ∎

In fact, we can show, using a similar proof, that the limit language of a Class One cellular automaton can be a non-context-free language or even a language of any given time or space complexity. It has been suggested that the limit languages are related to the classification. The above shows that there is no such simple relation. However, descendant languages of finite configurations are indeed related to our classification.

Theorem 3. *If a cellular automaton $A = (S, N, f)$ is in Class One, then there exists a stable state $s \in S$ such that the descendant language $D_s(c)$ is finite for each s-finite configuration c.*

Proof. By the definition of Class One automaton A started in any s-finite configuration reaches the homogeneous configuration (in s) in possibly unbounded but finite number of steps. ∎

Theorem 4. *Let $A = (S, N, f)$ be a cellular automaton in Class One. Then the global mapping of A is not injective (even restricted to finite configurations in the sense of [6]), i.e., there exists a Garden of Eden configuration [5].*

Proof. Obvious. ∎

Theorem 5. *It is undecidable whether all the finite configurations of a given cellular automaton evolve to the quiescent configuration (in linear time).*

Proof. We reduce the emptiness problem for Turing machines to this problem. Here, we use the Turing machine model with a single one-way infinite tape and a single read-write head. The reduction is not straightforward due to the fact that unlike a Turing machine, a cellular automaton does not have a separate input alphabet and thus has no distinguished initial configurations. If the finite configurations of a cellular automaton A are used to encode the instantaneous descriptions (IDs) of M, then the encoding of every ID of M including the unreachable ones can appear as an initial configuration of A.

For a given Turing machine M, we construct a cellular automaton A such that M accepts an empty language if and only if all finite configurations of A evolve to the quiescent configuration. Each state of A except the quiescent state consists of three components. Therefore, each segment can be viewed as having three tracks. The first track is used to store a sequence of IDs of M separated by dollar signs. The second track is a working track and is used to check whether the first track contains a legal sequence of IDs started with a valid initial ID and ended with an accepting ID. The third track contains a signal which controls the phase changes of the second track. A valid third track should be of the following form:

$$>> \ldots > S << l \ldots <$$

and its validity can be checked by a local function in one step.

A segment that has a valid third track and has two special boundary states at both ends evolves as follows: The first track remains unchanged if no quiescent state is generated within the segment. The second track works in three phases in turn. In Phase I it does the cleaning job, meaning that this track is being cleared cell by cell. In Phase II it simulates the Firing Squad Synchronization problem. In Phase III all cells of this track start at the same time. They check whether track 1 stores a legal sequence of IDs of M. The three phases take $n-1$, $2n-2$, and $n-1$ steps, respectively, where n is the length of the segment. The timing is controlled by the third track by moving a signal. Track 2 goes repeatedly through these three phases in turn. However, the quiescent state will be generated and will spread if

1. the segment has an invalid third track; or
2. the segment does not have two special boundary states at both ends; or
3. the second track finds, in its Phase III, that the first track does not contain a legal sequence of IDs of M.

If M accepts an empty language, then no legal sequence of ID's exists. All finite configurations of A evolve to the quiescent configuration in linear time. If the language accepted by M is not empty, then all segments that encode legal accepting sequences of M will never evolve to the quiescent configuration. Since the emptiness problem for Turing machine is undecidable in general, our problem is also undecidable.

Note that an initial segment of A may contain "junk" in the second track that may be in any phase. Therefore, a segment of length n that eventually evolves to the quiescent configuration may take up to $8n - 9$ steps to be totally quiescent. ■

Corollary 1. *It is undecidable whether a given cellular automaton is in Class One.*

Proof. By using the above theorem and considering the stable state to be the quiescent state. ■

We also have the following by-products.

Corollary 2. *Let M be an arbitrary Turing machine and Ω be the set of all the instantaneous descriptions (IDs) of M. Then it is undecidable in general whether M starting with $c \in \Omega$ halts for all $c \in \Omega$.*

Proof. Instead of reducing directly the halting problem for Turing machines to the new problem, we reduce the problem from theorem 5. We have not found a proof that uses the former approach without actually mimicking the latter.

For any cellular automaton A we construct a Turing machine M as follows. Informally, M simulate the mapping defined by f. It stops if all the nonblank symbols (states of A) on the tape become quiescent. So, all the finite configurations of A evolve to the quiescent configuration if and only if M starting with any ID halts. The latter is undecidable because of the undecidability of the former by theorem 5.

A formal construction of M is given in the appendix. ■

Corollary 3. *Let $M = (Q, \Gamma, \Sigma, \delta, q_0, q_f, B)$ be a Turing machine with the special property of $\Gamma = \Sigma \cup \{B\}$. Then it is undecidable in general whether M halts on all inputs.*

4. Class Two

We define Class Two cellular automata as follows.

Definition 2. *A cellular automaton A is said to be in Class Two if there exists a stable state s such that every s-finite configuration of A has an ultimately periodic evolution.*

Obviously, Class One is included in Class Two. Our definition for Class Two appears to specify a larger class than that of Wolfram's. However, for the cellular automata with $k = 2$, $r = 2$, our definition again coincides with his.

Example 4. *The cellular automaton with $k = 2$, $r = 2$ and function code 52 is not in Class Two since the finite configuration*

1101111001000010011110 11

does not have a ultimately periodic evolution.

Similarly, like for Class One we have the following.

Theorem 6. *Let A be in Class Two. Then there exists a stable state s such that the descendant language of every s-finite configuration is finite.*

Theorem 7. *If A is in Class Two with respect to a stable state s and its global mapping is surjective, then every s-finite configuration has a periodic evolution.*

Proof. The surjectivity of the global mapping G_f implies that G_f is injective on s-finite configurations. Assume that there exists an s-finite configuration c such that the evolution of c is not periodic. Since the evolution of c is ultimately periodic, there exist two s-finite configurations that are mapped to the same configuration. This contradicts the condition that G_f is injective on all s-finite configurations. ∎

Theorem 8. *It is undecidable whether a cellular automaton is in Class Two.*

Proof. We use the result in corollary 2. We show that for any Turing machine M we can construct a cellular automaton A such that every finite configuration of A has an ultimately periodic evolution if and only if M starting with any ID halts.

We define a valid segment of A to be one that encodes a valid ID of M and has two boundary states l and r at the left and the right end, respectively. A valid ID of M is a string of uqv where u and v are strings of tape symbols and q is a state. It means that M is in state q, the nonempty portion of its tape is uv and the reading head is over the first symbol of v. In order to make the validity of an encoding checkable by a local CA function, all the tape symbols to the left of the head of M should have a direction bit '$\rightarrow$' and all the ones to the right of the head should have a bit '$\leftarrow$'. If a segment is *invalid*, it must have some *invalid neighborhood*.

The successor of a segment in the evolution of A is defined as follows. The successor of a *valid* segment, which encodes an ID I of M, encodes an ID I' such that I' is the ID computed from I by M and has its two boundaries expanded by one cell in both directions. However, if ID I does not have a successor ID, the quiescent state is generated at the position of the head. The successor of an *invalid* segment contains quiescent cells generated by invalid neighborhoods of its predecessor. The quiescent state is spreading at a speed of two cells at each step in both directions. It overruns all states except the left boundary state l from the left and the right boundary state r from the right.

If there exists an ID d of M such that M starting with d never halts, then the finite configuration containing a single segment that encodes d does not have an ultimately periodic evolution, and thus it is not in Class Two. If there is no such ID at all, then every finite configuration of A evolves to the quiescent configuration. Thus A is in Class Two. ∎

Theorem 9. *It is undecidable whether a Class Two cellular automaton is in Class One.*

Proof. This has actually been proved in the proof of theorem 5. ∎

5. Class Three and Four

Definition 3. *A cellular automaton is in Class Three if there exists a stable state s such that for any pair of s-finite configurations c_1 and c_2, it is decidable whether c_1 evolves to c_2. Class Four includes all cellular automata.*

Clearly, Class One and Class Two are contained in Class Three. We can also show that the cellular automata with $k = 2$, $r = 2$, and function code 2, 6, 10, 12, 14, 18, 22, 26, 28, 30, 34, 38, 42, 44, 46, and 50 are all in Class Three by using their property of expanding. However, we are unable to show that the ones with function code 20 and 52 are or are not in Class Three. One way of showing that a cellular automaton is not in Class Three is to show that this cellular automaton is capable of simulating a universal Turing machine.

Theorem 10. *A cellular automaton that simulates the computation of a universal Turing machine is not in Class Three. A universal cellular automaton is not in Class Three.*

Proof. Obvious. ∎

See [1] for a simple universal cellular automaton.

Theorem 11. *A cellular automaton is in Class Three if and only if there exists a stable state s such that the descendant language of every s-finite configuration is recursive.*

Proof. The theorem is just a restatement of definition 3. ∎

A cellular automaton with the property that every finite configuration either evolves to the quiescent configuration or evolves to a longer finite configuration in a bounded time is clearly in Class Three.

The next theorem appears to be obvious. However, our proof is more complicated than we expected. We encourage reader to give a simpler proof.

Theorem 12. *It is undecidable whether a given cellular automaton is in Class Three.*

Proof The idea of our proof is the following. For any given Turing machine M (with a single head and a one-way infinite tape), we construct a cellular automaton $A(M)$ as follows. Each segment of nonquiescent cells of $A(M)$ has two tracks. One track is used to simulate the computation of M. The other is used to simulate a universal Turing machine U. The construction of $A(M)$ should guarantee that if M halts on input ε, then every finite configuration of $A(M)$ eventually evolves periodically or to the quiescent configuration. Otherwise, $A(M)$ can simulate all the computations of U. Therefore, by the definition of Class Three, M halts on ε if and only if $A(M)$ is in Class Three. The undecidability of the former implies the undecidability of the latter. Thus, we have proved the theorem. More details follow. See figure 1.

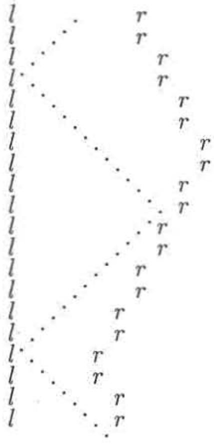

Figure 1: The evolution of a valid segment.

A valid segment of $A(M)$ has two boundary states l and r. The cells between the two boundaries have two tracks. The second track contains an encoding of an instantaneous description (ID) of U. See [3, page 148] for the definition of an ID. One computation step of U is simulated by two evolution steps of $A(M)$. The boundaries of a segment are the boundaries of the ID of U on its second track. On the first track, a signal travels between two boundaries at a speed of one cell each step. Whenever the signal reaches the left boundary, it turns to the right and a simulation of M is initiated on this track and proceeds in the area between the left boundary and the signal. If a segment is invalid or the simulation of M or U halts, quiescent states are generated and they spread in both directions until all the cell become quiescent. ∎

6. Products of cellular automata

Definition 4. *Let $A_1 = (S_1, N_1, f_1)$ and $A_2 = (S_2, N_2, f_2)$ be two cellular automata of a same dimension with q_1 and q_2 being the quiescent state of A_1 and A_2, respectively. Then the product of A_1 and A_2 is the cellular automaton $A = (S, N, f)$, where*

$$S = \{\langle s_1, s_2\rangle | s_1 \in S_1, s_2 \in S_2\},$$

$$N = N_1 \cup N_2,$$

the quiescent state is the state $\langle q_1, q_2\rangle$, and A can be considered having two tracks, f simulates f_1 on the first track and f_2 on the second track.

Theorem 13. *Let A_1 and A_2 be two cellular automata in Class I_1 and Class I_2, respectively. Then the product of A_1 and A_2 is in Class max(I_1, I_2).*

Proof. Obvious. ∎

In [9] Wolfram discussed how a change of one site (cell) in the initial configuration propagates in the development of that configuration. He observed that the propagation of the difference is typical for each of his classes. We obtain the same result as an application of theorem 13.

Let two cellular automata have the same set of states. Starting from the same configuration, these two cellular automata may evolve differently. In order to see the difference of these two evolutions, we can define a morphism h on the product of the two automata such that

$$h : \langle p, q\rangle \longrightarrow 1, \text{ if } p \neq q;$$

$$h : \langle p, q\rangle \longrightarrow 0, \text{ if } p = q.$$

Now, in order to apply theorem 13 to the problem of propagation of a single site difference, we consider the product of the given CA A with itself and run this automaton on an initial configuration c in which both tracks agree everywhere but in one cell. Clearly, the product of A with A belongs to the same class of A and if we observe the output h for computation initiating in c, we see that it corresponds (as the output is concerned) to the computation from one site and therefore the growth of non-zero sites indicates the propagation of difference. Hence, we can conclude that the propagation of difference has the same "class characteristics" as growth from one or finite number of non-zero sites.

Appendix A. Formal construction of M in the proof of corollary 2

Let $A = (S, N, f)$ be a linear cellular automaton. Without loss of generality we assume that $N = (-h+1, -h+2, \ldots, -1, 0)$. We construct a Turing machine $M = (Q, \Gamma, \Sigma, \delta, q_0, q_f, B)$ as follows:

$\Gamma = S \cup \{B\}$,

$\Sigma = S$,

$Q = \{\langle s_1, s_2, \ldots, s_h\rangle \mid s_1, s_2, \ldots, s_h \in S\} \cup \{q_e, s_b, q_f\}$,

$\delta(\langle s_0, s_1, \ldots, s_{h-1}\rangle, s_h) = (\langle s_1, s_2, \ldots, s_h\rangle,\ f(s_1, s_2, \ldots, s_h), R)$, if s_0, s_1, $\ldots$, s_{h-1}, $s_h \in S$;

$\delta(\langle s_0, s_1, \ldots, s_{h-1}\rangle, B) = (< s_1, s_2, \ldots, s_{h-1}, q\rangle,\ f(s_1, s_2, \ldots, s_{h-1}, q), R)$, where q is the quiescent state of A, if $s_0, s_1, \ldots, s_{h-1} \in S$, and $s_i \neq q$ for some $i \in \{0, 1, \ldots, h-1\}$;

$$\delta(\langle q, q, \ldots, q\rangle, B) = (q_e, B, L);$$

$$\delta(q_e, q) = (q_e, B, L);$$

$$\delta(q_e, s) = (q_b, s, L), \text{ for all } s \in S - \{q\};$$

$$\delta(q_b, s) = (q_b, s, L), \text{ for all } s \in S;$$

$$\delta(q_b, B) = (\langle q, q, \ldots, q\rangle, B, R); \text{ and}$$

$$q_0 = \langle q, q, \ldots, q\rangle.$$

Intuitively, M repeatedly moves from left to right and from right to left. During its left or right move, it simulates the local mapping f on its input. Each state of M has h registers memorizing the last h symbols it has read. Note that the quiescent state q of A is not the blank symbol B of M. M starts at the left end of the input string with the state $q_0 = \langle q, q, \ldots, q\rangle$. When it reaches the right end of the input, it continues to read h Bs (considering them to be q for the mapping) and turns back to the left. During its right to left move, it changes all the trailing q's to Bs, and it starts a new cycle again when it meets the first B left to the nonblank string.

Appendix B. Example 3, continued

The transition table of CA from example 3 is given in figure 2.

	$f(x,y,z)$															
x	y z															
	00	01	0-	0q	10	11	1-	1q	-0	-1	--	-q	q0	q1	q-	qq
0	0	0	-	-	1	1	0	0	1	1	0	0	q	q	q	q
1	0	0	-	-	1	1	0	0	1	1	0	0	q	q	q	q
-	0	0	-	-	1	1	0	0	1	1	0	0	q	q	q	q
q	q	q	q	q	1	1	0	0	q	q	q	q	q	q	q	q

Figure 2: The transition table of the CA from example 3.

References

[1] J. Albert and K. Culik II, "A Simple Universal Cellular Automaton and its One-Way and Totalistic Version", *Complex Systems*, **1** (1987) 1–16.

[2] K. Culik, J. Pachl, and S. Yu, "On the Limit Sets of Cellular Automata", *Research Report* CS-87-47, Department of Computer Science, University of Waterloo (1987).

[3] J. E. Hopcroft and J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation*, 1979.

[4] L. P. Hurd, "Formal Language Characterization of Cellular Automaton Limit Sets", *Complex Systems*, **1** (1987) 69–80.

[5] E. F. Moore, "Machine Models od Self-Reproduction", *Proc. of Symposium in Applied Mathematics*, **14** (1961) 17–33.

[6] D. Richardson, "Tessellation with Local Transformations", *Journal of Computer and System Sciences*, **6** (1972) 373–388.

[7] S. Wolfram, "Universality and Complexity in Cellular Automata", *Physica*, **10D** (1984) 1–35.

[8] S. Wolfram, "Computation Theory of Cellular Automata", *Communications in Mathematical Physics*, **96** (1984) 15–57.

[9] S. Wolfram, "Twenty Problems in the Theory of Cellular Automata", *Physica Scripta*, **T9** (1985) 170–183.