Complex Systems **1** (1987) 1–16

A Simple Universal Cellular Automaton and its One-Way and Totalistic Version

Jürgen Albert
Lehrstuhl für Informatik II,
Universität Würzburg,
D-800 Würzburg, Germany

Karel Culik II
Department of Computer Science,
University of Waterloo,
Waterloo, Ontario N2L 3G1, Canada

Abstract. In this paper we first derive two normal form constructions for cellular automata to transform any given (one-dimensional) cellular automaton into one which is one-way and/or totalistic. An encoding of this restricted type of automaton together with any initial configuration becomes the input of our small universal cellular automaton, using only 14 states. This improves well-known results obtained by simulation of small universal Turing machines and also some recent results on universal totalistic cellular automata.

1. Introduction

Interest in cellular automata (CA) has been renewed since their application to the study of complex systems [17,18,19]. In this context the universality of CA was discussed in [19] and open problems about universal CA with a small number of states were stated in [20]. We give some results here.

A cellular automaton is said to be *universal* if it can simulate every Turing Machine (TM) or, even stronger, if it can simulate every CA of the same dimension. The first universal cellular automaton was given in the famous work of J. von Neumann on the simulation of self-reproduction. He gave a 2-dimensional universal and self-reproducing CA with 29 states. This was improved to 20 states in [1].

Also well known is the work on small universal Turing Machines (cf. [9,10,13]). In this case the goal is to minimize simultaneously the number of states and the number of tape symbols. In [10] it is shown that there is a universal Turing Machine with either 4 symbols and 7 states or with 6 symbols and 6 states. Smith [14] has shown that any Turing Machine with m symbols and n states can be simulated by a one-dimensional CA whose

uniform cells has $m + 2n$ states. Thus either one of the universal TM from [10] yields CA with $4 + 2 \times 7 = 6 + 2 \times 6 = 18$ states. Here we improve this result and show that there is a universal one-dimensional CA with 14 states. We believe that some more effort in the detailed "low-level programming" of our basic strategy can decrease this by 2 or 3 states. However, the minimal number of states of a universal one-dimensional CA could still be much smaller, since CA with 3 or 4 states already show astonishingly complex behaviour.

The well known "Game of Life" is a "semi-totalistic" CA which means that the next state of a cell only depends on its own current state and the sum of the states of its neighbors. S. Wolfram also introduced an even more restricted type of CA, called totalistic. The next state of a cell of such CA depends only on the sum of all the states in its neighborhood, including its own. D. Gordon [7] has shown that totalistic one-dimensional CA can simulate every Turing Machine and we will strengthen this result and show that every one-dimensional CA can be simulated by a totalistic one. We will use this result in the construction of our small universal CA. We will also use the result that every one-dimensional CA can be simulated by a one-way (unidirectional) one-dimensional CA [4,5,16] and that one-way CA are equivalent to trellis automata [2].

We will show our results for CA with the transition function dependent on the state of each cell and its immediate left and right neighbors ($r = 1$ in [19]). However, our results can easily be extended to larger neighborhoods.

2. Preliminaries

As we will consider only one-dimensional and homogeneous cellular automata in this paper we will restrict ourselves to this special case in the following definitions. For more general terminology see [3] or [15].

Intuitively, a cellular automaton consists of a doubly infinite arrray of cells. All cells are identical copies of one single finite automaton. The local transition function of each cell only depends on the actual states of its left and right neighbor and itself. Thus, a computation of a cellular automaton can be defined in the straightforward way as the synchronous application of the local transition function at each level. The set of states always contains a so-called quiescent state q with the property: if a cell and its left and right neighbors are quiescent at time t then this cell is quiescent at time $t+1$. As we assume finite output for the cellular automaton this implies that there is a finite number of nonquiescent cells in the initial configuration and in every subsequent configuration as well.

Definition 1. *A (one-dimensional, homogeneous) cellular automaton is a triple $A = (Q, d, q)$ where Q is a finite set of states, d is the local transition function, $d : Q \times Q \times Q \to Q$ where the arguments of d are used in the following meaning d(state of left neighbor, own state, state of right neighbor), and q in Q is the quiescent state, i.e. it holds $d(q, q, q) = q$.*

Definition 2. *A configuration of A is a mapping $C : Z \to Q$, where Z denotes the set of integer numbers such that $C(i) = q$ (quiescent state) for all but finitely many i's. The set of nonquiescent cells of a configuration C is called the support of C.*

Definition 3. *A computation of A is a sequence of configurations*

$$C_0, C_1, C_2, \ldots, C_n, \ldots$$

where C_0 is the initial configuration and each configuration C_{i+1} is generated by simultaneous invocation of the transition function d for all cells of A in configuration C_i.

3. Simulation by one-way automata

For our construction of a universal cellular automaton in subsequent chapters we will need an automaton which is totalistic and one-way.

A cellular automaton A with set of states Q and transition function $d : Q \times Q \times Q \to Q$ is called one-way, if d only depends on the state of the own cell and the state of its right neighbor cell. Thus we can write the transition function of a one-way cellular automaton in the form $e : Q \times Q \to Q$ with the convention that the arguments of e consist of the states of the own cell and those of its right neighbor.

In this section we will outline a transformation of an arbitrary cellular automaton to a one-way cellular automaton in order to specify bounds for the increase in the number of states and in time-steps needed in the simulation.

In [16] a similar technique was used for the case of real-time cellular automata, more general cases were considered in [4] and [5].

The construction of the one-way automaton can be described briefly by:

1. merging of each state with the state of its right neighbor
2. shifting to the left during the state transition.

Let $s_1, s_2, \ldots, s_k$ be states in a configuration, such that $s_1 \neq q$ and let $t_0, t_1, \ldots, t_{k+1}$ be the states after one transition as shown below

$$\begin{array}{cccccccccccc} q & q & q & s_1 & s_2 & s_3 & \ldots & s_k & q & q & q \\ q & q & t_0 & t_1 & t_2 & t_3 & \ldots & t_k & t_{k+1} & q & q \end{array}$$

Then it takes two time-steps in our simulating automaton to produce the transition:

$$\begin{array}{cccccccccccc} q & q & q & s_1 & s_2 & s_3 & \ldots & s_k & q & q & q \\ q & t_0 & t_1 & t_2 & t_3 & \ldots & t_{k+1} & q & q & q & q \end{array}$$

For the given cellular automaton A let Q be the set of states and $d : Q \times Q \times Q \to Q$ the transition function. As defined above, in the simulating one-way cellular automaton A' the transition function d' will consider only the states of its own cell and its right neighbor cell.

Let $Q' = Q \cup Q \times Q$ and define $d' : Q' \times Q' \to Q'$ as follows

$$\begin{aligned} d'(u,v) &= uv \text{ for } (u,v) \neq (q,q) \\ d'(q,q) &= q \\ d'(uv,vw) &= d(u,v,w) \\ d'(q,qu) &= d(q,q,u) \\ d'(wq,q) &= d(w,q,q) \end{aligned}$$

for all u, v, w in Q.

Thus, the transition shown above is completed in two time-steps by A' in the following way

$$\begin{array}{cccccccc} q & q & s_1 & s_2 & \dots & s_{k-1} & s_k & q & q \\ q & qs_1 & s_1s_2 & s_2s_3 & \dots & s_{k-1}s_k & s_kq & q & q \\ t_0 & t_1 & t_2 & t_3 & \dots & t_k & t_{k+1} & q & q \end{array}$$

The details of this construction are straightforward, so we can state the following theorem:

Theorem 1. *For every cellular automaton A with k states there exists a one-way cellular automaton A' which simulates A twice slower and A' needs at most $k^2 + k$ states.*

In the above construction of the simulating one-way cellular automaton one can delay the given transitions by "aging" the states in Q (e. g. $a, a', a'', a''', \dots$) to obtain slower expansion of the nonquiescent cel Therefore it is clear that the following corollary holds:

Corollary 1. *For every cellular automaton A with k states and for every $m \geq 1$ there exists a one-way cellular automaton A' which simulates A $(m+1)$-times slower and A' needs at most $k^2 + mk$ states.*

Later, we will have to apply this corollary for $m = 3$ in the construction of a universal one-way cellular automaton.

4. Simulation by totalistic automata

In this section we will show that each cellular automaton can be simulated (without loss of time) by a cellular automaton which has a totalistic transition function and uses up to four times as many states as the original automaton.

Definition 4. *A cellular automaton A with set of states Q and transition function $f : Q \times Q \times Q \to Q$ is called totalistic, if $Q \subseteq N$ (non-negative) and there exists a function $f' : N \to N$ such that $f(x,y,z) = f'(x+y+z)$ for all x, y, z in Q.*

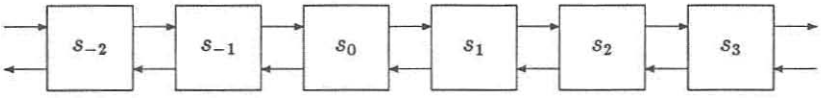

Figure 1: A configuration of cellular automaton A.

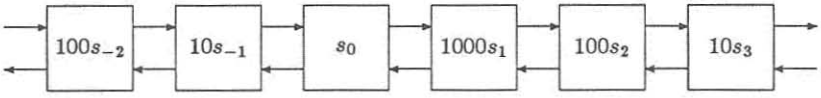

Figure 2: A configuration of the totalistic CA simulating CA A.

The transformation of an arbitrary cellular automaton whose states are identified with non-negative integers is now accomplished by a cyclic "coloring" of cells. We use four different powers of a basis such that in the number representation of the sum of three neighboring states left neighbor, right neighbor and own cell are still identifiable by the position of a missing entry.

Consider, for example, a cellular automaton A with set of states $Q = 1, 2, \ldots, n$ and transition function $d : Q \times Q \times Q \to Q$ and let the Figure 1 depict part of configuration of A.

Let $B = n + 1$ be the basis for the coloring factors 10, 100 and 1000 (in B-ary notation). Then the configuration in Figure 1 changes to the one in Figure 2.

Thus our new set of states is

$$Q' = \{sm \mid s \in Q, m \in \{1, 10, 100, 1000\}\}.$$

So Q' contains $4n$ states and we can define now the (partial) totalistic transition function $d' : N \to Q'$ as follows

$$\begin{aligned} d'(xyz) &= 10d(x,y,z) \\ d'(xyz0) &= 100d(x,y,z) \\ d'(yzx) &= 1000d(x,y,z) \\ d'(z0xy) &= d(x,y,z) \end{aligned}$$

for all x, y, z in Q.

Again $xyz, xyz0, yz0x, z0xy, 10, 100, 1000$ are to be interpreted as numbers in the B-ary system.

It is now straightforward, that the cellular automaton A with set of states Q' and totalistic transition function d' correctly simulates A without loss of time and it therefore holds

Theorem 2. *For every cellular automaton A there exists a totalistic cellular automaton A' which simulates A without loss of time and has at most four times as many states as A.*

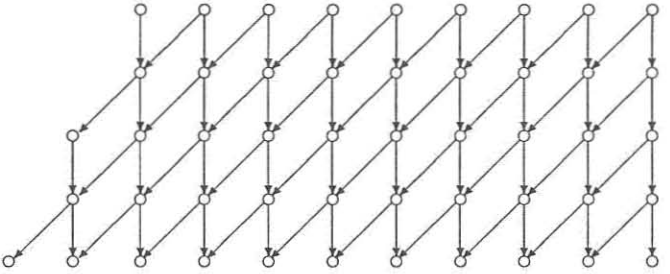

Figure 3: The unrolling (time-space diagram) of a one-way CA.

If only one-way cellular automata are considered, the above construction can be simplified using only three different powers of B for the coloring of the cells, whereby the one-way property of the given automaton is preserved.

Corollary 2. *For every one-way cellular automaton A there exists a one-way totalistic cellular automaton A' which simulates A without loss of time and has at most three times as many states as A.*

For the more general case of an arbitrary (regular) systolic network this technique of coloring its underlying graph is used in [6] to show that actually every systolic network can be transformed into a totalistic one.

5. Informal description of construction

In the previous section we have shown that any CA can be simulated by one which is one-way and totalistic and such that the expansion of the nonquiescent part to the left can be delayed by any constant factor m. We choose $m = 3$ to achieve expansion to the left at most at half-speed.

Our universal automaton U will simulate any given one-way totalistic cellular automaton A (which expands at most at half speed to the left) with any given initial configuration. We will encode this pair as initial configuration of the universal automaton U. A constant number (depending only on the number of states of A) of steps of U is needed to simulate one step of A. Four steps of the time-space diagram (unrolling) of an one-way automaton with a starting configuration of length 9 is shown in Figure 3. This unrolling is obviously computationally equivalent to the trellis structure shown in Figure 4. We will use this observation in our construction of the universal cellular automaton U. Note that the computations which took place in one cell of the first automaton and therefore being depicted in vertical lines in the first figure, are now shifting to the right in each step.

Example: Let A be a totalistic one-way cellular automaton with set of

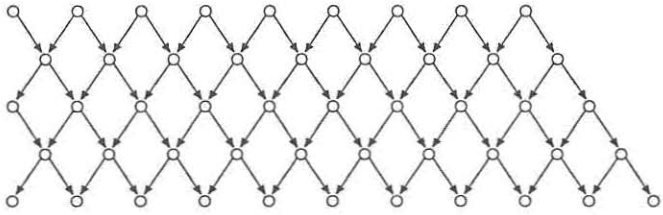

Figure 4: A trellis structure isomorphic to the encoding of a one-way CA.

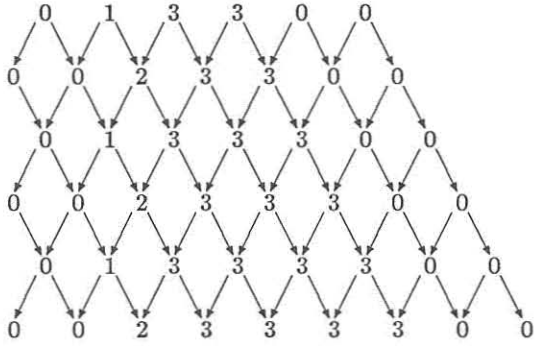

Figure 5: A computation of one-way CA A.

states $Q = 0, 1, 2, 3, 4$ and transition function d given by

i	0	1	2	3	4	5	6
$d(i)$	0	0	1	3	2	3	3

Since A is one-way the nonquiescent states expand to the right only. An example of a computation of A transformed into the trellis shape is shown in Figure 5. In the simulation of the automaton A each of the nonquiescent cells of A will be encoded as a block of constant size of cells in U. This block contains the given transition-table for the cells of A, and by the marking of the appropriate table-entry also the encoding of the current state of a cell, i.e. if a transition $d(i+j) = k$ takes place in a cell of A, in the corresponding block the $(i+j)$th table-entry — which holds an encoding of the number k — becomes marked.

So in our automaton U the main actions to simulate one time-step if the given automaton A will be grouped into three phases:

Phase 1: Send information about current state to left neighbor block.

Phase 2: Send information about current state to right neighbor block.

Phase 3: Process information to look up next state.

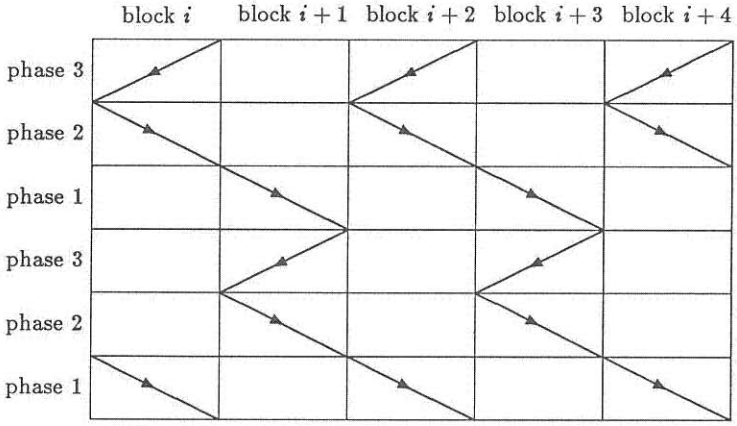

Figure 6: The flow of signals in the adjacent blocks of the universal CA U.

6. Outline of construction

As shown above the next states are always computed between the two cells whose states are to be considered for the table-lookup. Therefore it is convenient to use alternatively "activated" blocks and "empty" blocks and to change the roles of both in the odd and even steps (beats) of the computation. This is achieved by sending messenger signals over these blocks where the signals have to complete various tasks during the three phases mentioned above.

Phase 1: Start from the right end of an active block and find a table-entry representing the current state; activate a number of signals according to the current state which are moving to the empty block to the left; continue to move to the left end of the block.

Phase 2: Bounce at the left end of the block and find once more the table-entry representing the current state; accordingly activate signals which are now moving to the empty block to the right; while clearing all temporary marks continue to move to the right end of the block, leaving an empty block behind.

Phase 3: Cross over to the block to the right; while moving to the right end of this block process the signals which have been accumulated here during the previous phases; bounce at the right end.

The flowing of messenger signals over some adjacent blocks during the phases of two consecutive beats is illustrated in Figure 6.

Furthermore we have to take care that after each beat a new block at the righthand end of the nonquiescent part can be activated to simulate a possible expansion to the right. This is achieved by copying (at least) one

complete block to the right during each beat and by activating one new block which starts in the encoding of the quiescent state.

By our construction we can assume that there is no expansion to the left and so we only need to activate the leftmost block at each second beat. This is realized by a special block and messenger signals.

Details of these constructions and encodings, the transition-table for U and examples are found in the appendix.

Theorem 3. *There exists a universal automaton U with 14 states which can simulate any given totalistic one-way cellular automaton A with any initial configuration.*

In [14]:Theorem 4 it is shown that any Turing machine T with m tape symbols and n states can be simulated by a cellular automaton with $m+2n$ states. Thus either of the universal Turing machines with 6 symbols and 6 states or with 4 symbols and 7 states ([10]) yields a universal cellular automaton with 18 states.

We can now apply the normal form construction of our previous sections to the universal cellular automaton U of Theorem 6.1 to generate universal cellular automata which are also one-way and/or totalistic.

Corollary 3. *For every universal cellular automaton U with n states there exists a universal one-way cellular automaton U_1 with $n^2 + n$ states.*

Corollary 4. *For every universal cellular automaton U with n states there exists a universal totalistic cellular automaton U_2 with $4n$ states.*

Corollary 5. *For every universal cellular automaton U with n states there exists a universal one-way totalistic cellular automaton U_3 with $3(n^2 + n)$ states.*

Acknowledgements

This research was supported by the Natural Sciences and Engineering Research Council of Canada under Grant No. A-7403.

Appendix

For the encoding of the given transition-table as a block of cells in U we can assume without loss of generality that for the given totalistic one-way cellular automaton A the set of states Q is defined as $Q = \{3, 4, 5, \ldots, k\}$, where 3 is identified with the quiescent state in A.

Each block consists now of a sequence of $2k + 1$ sub-blocks of the form

$$\#z\underbrace{000\ldots000}_{i-times}\underbrace{111\ldots111}_{j-times}$$

and a block-endmarker #, where z is in $\{0,1\}, i+j=2k+1$, and if in the given transition function d we have $d(m)=n$ then in the mth sub-block of each block it holds $j=n$. For all other sub-blocks we have $j=3$. Thus a block consists of $(2k+2)^2$ cells.

If a cell of the original one-way totalistic automaton is executing the transition $d(r)=s$, then in the corresponding block of U exactly in the first r sub-blocks from the left we have $z=1$ and in all other sub-blocks $z=0$.

The computation of the next transition in the simulating automaton U then takes $3(2k+2)^2+7(2k+1)$ times steps which constitute one "beat" of U.

Example: Let $Q=3,4,5$ be the set of states and the transition-function d be defined as

i	6	7	8	9	10
$d(i)$	3	4	4	5	4.

Then a transition by $d(9)=5$ corresponds to the following situation in the block at the beginning of the beat:

```
#100000000111#100000000111#100000000111#100000000111#100000000111
#100000000111#100000001111#100000001111#100000011111#000000001111
#0000000001aa# .
```

Here *aa* at the end of the block shows that this block is activated and that the new state 5 has to be sent to left and right neighbors.

It can also be seen easily that the transition-function has been extended to

i	1	2	3	4	5	6	7	8	9	10	11
$d(i)$	3	3	3	3	3	3	4	4	5	4	3

An inactive, "empty" block then looks like

```
#000000000111#000000000111#000000000111#000000000111#000000000111
#000000000111#000000001111#000000001111#000000011111#000000001111
#000000000111#.
```

If a block is activated at the beginning of the beat, his right neighbor block will become activated at the next beat. In order to be able to expand, it is therefore necessary to start a new activation from the left end at each second beat. This is done simply by sending a synchronizing signal up and down over a sufficiently long sequence of 1's. The first block after this synchronizing left part contains only table-entries representing the quiescent state. So each second beat the "quiescent block" at the left end is activated and no expansion to the left can occur.

At the right end copying of a block is performed by pushing an encoded form of a block to the right and simultaneously leaving an empty block behind, which will be activated later as representing the original quiescent

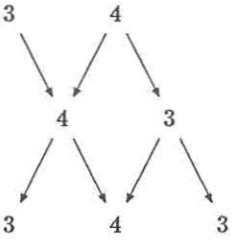

Figure 7: A computation of one-way CA B.

```
qqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqq1111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111a11111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111111#
1#00011111111#00000111#0000000111#0000000111#0000000111#0000000111
#0000000111#0000000111#0000000111##1000000111#1000000111#100000011
1#1000000111#1000000111#1000000111#0000000111#0000000111#00000001a
a##0000000111#0000000111#0000000111#0000000111#0000001111#00000001
11#0000001111#0000001111#0000000111##1000000111#1000000111#1000000
111#1000000111#1000001111#0000000111#0000001111#0000001111#0000000
1aa##1000000111#1000000111#1000000111#1000000111#1000001111#100000
0111#1000001111#1000001111#1000000111##1000000111#1000000fg1g0h0h0
h0g1g1h0hhhhhggg1h0h0h0g1g1g0hh0h0hgggghhh0h0g1g1g0h0h0h1gg1qqqqqq
```

Figure 8: The initial configuration of U which encodes the configuration 34 of B.

state. During one beat a little more than a complete block is created at the right, but since activation of blocks is initiated from the left — synchronized with beat — this does not affect the simulation of the nonquiescent part of the original totalistic one-way automaton.

Example: Consider CA B given by the following transition-table

i	1	2	3	4	5	6	7	8	9
$d(i)$	3	3	3	3	4	3	4	4	3

Three consecutive configurations of B are shown in Figure 7. The three snapshots of U corresponding to these three configurations are shown in Figures 8, and 9. In this case one beat of U consists of 363 single transition-steps. Figure 8 shows the initial configuration of U, the configurations after the first beat (363 steps), and after the second beat (726 steps) are in Figure 9.

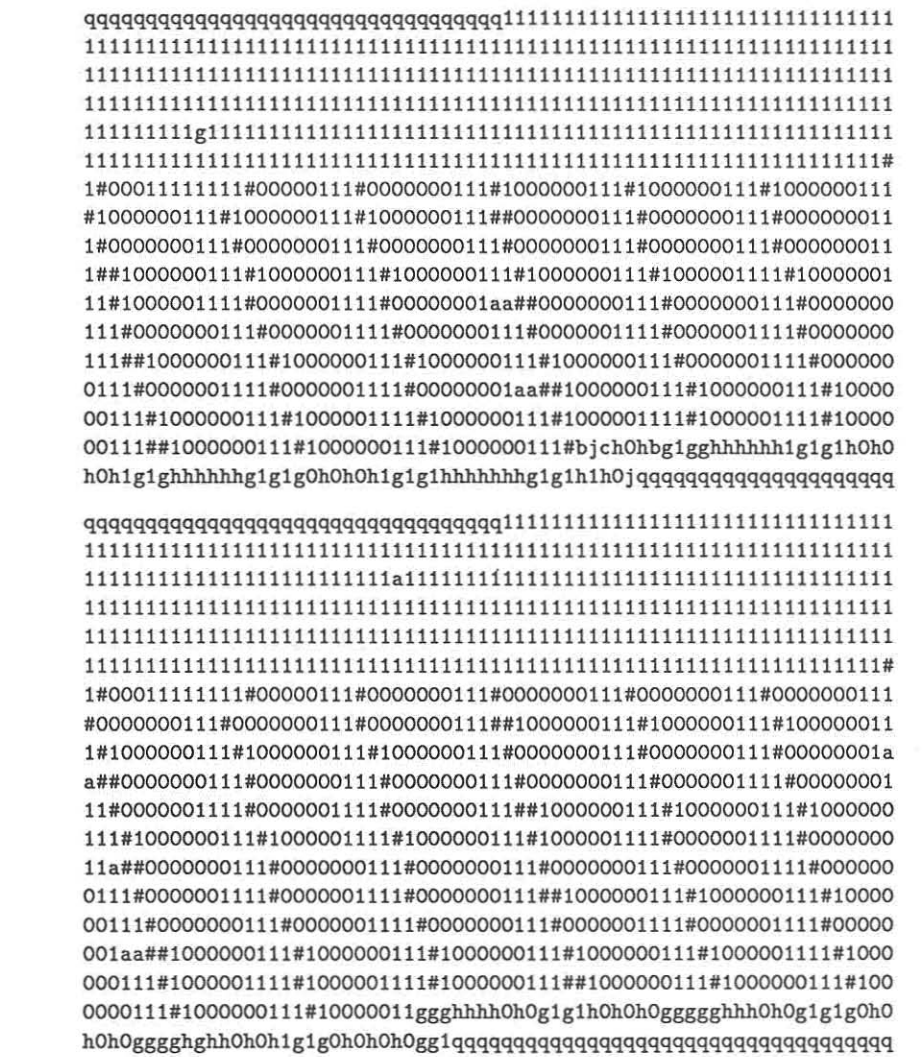

Figure 9: The configuration of U after the first beat (363 steps) and after the second beat (726 steps), which encode configurations 43 and 343 of B.

	#	0	1	a	b	c	d	e	f	g	h	i	j	q
#	#	-	-	1	-	-	-	1	-	-	h	1	-	-	-
#	0	#	0	1	f	1	0	b	-	1	b	0	#	j	-
#	1	#	-	-	-	-	#	-	-	-	-	-	#	q	-
#	a	#	-	-	-	-	#	-	-	-	-	-	-	-	-
#	b	#	b	a	1	d	-	-	-	a	-	0	-	j	-
#	c	#	g	-	-	-	-	-	-	-	-	-	-	0	-
#	d	-	-	-	-	-	d	-	-	-	-	-	-	-	-
#	e	a	-	a	-	-	-	-	-	-	-	-	-	-	-
#	f	c	-	-	-	-	-	-	g	-	-	-	-	-	-
#	g	#	-	-	b	#	-	-	-	-	-	-	-	-	j
#	h	#	0	1	e	h	-	-	-	-	0	1	#	j	-
#	i	-	-	1	-	-	c	-	-	-	-	-	-	-	-
#	j	#	a	-	b	1	-	-	-	-	-	-	-	-	-
0	0	-	0	-	-	0	0	-	0	-	1	0	-	j	-
0	1	-	0	1	1	0	0	1	0	1	1	0	-	j	-
0	a	-	b	a	a	b	-	-	-	-	1	0	-	-	-
0	b	-	b	-	-	b	-	-	-	-	-	0	-	j	-
0	c	-	c	-	-	-	-	-	-	-	-	-	-	0	-
0	d	-	c	d	-	-	-	-	-	-	-	-	-	-	1
0	f	-	0	1	-	-	-	-	-	-	-	-	-	-	-
0	g	-	0	1	g	h	-	-	-	g	1	0	-	0	q
0	h	-	0	-	-	h	-	-	-	-	1	0	-	j	-
0	j	-	0	-	-	-	-	-	-	-	-	-	-	-	-
0	q	-	0	1	-	-	-	-	-	-	1	0	-	0	q
1	#	#	-	1	1	-	#	-	g	e	h	1	#	-	-
1	0	#	0	1	-	0	-	-	0	-	1	0	#	j	-
1	1	#	0	1	1	-	#	1	-	1	1	0	i	-	-
1	a	#	-	a	a	-	#	-	-	-	1	-	-	-	-
1	b	#	b	-	-	b	c	-	-	-	a	-	#	-	-

	#	0	1	a	b	c	d	e	f	g	h	i	j	q
c	1	#	0	-	-	-	-	-	-	-	-	1	0	-	-
c	a	-	-	-	-	-	-	-	-	-	-	-	0	-	-
c	b	-	-	-	-	-	-	-	-	-	-	-	0	-	-
c	d	-	-	-	-	-	g	-	-	-	-	-	-	-	-
c	f	c	-	-	-	-	-	-	-	-	-	f	-	-	-
c	g	-	-	1	-	-	-	-	-	-	-	-	-	-	-
c	h	-	-	-	-	-	-	#	-	-	-	-	-	-	-
c	i	-	-	1	-	-	-	-	-	-	-	-	-	-	-
d	#	-	-	h	-	-	-	-	-	-	-	-	-	-	-
d	0	d	0	-	-	-	-	-	-	-	-	h	#	-	-
d	g	-	-	g	-	-	-	-	-	-	-	g	h	-	-
d	h	i	-	g	-	-	-	-	-	-	-	g	h	-	-
e	#	i	-	-	-	-	-	-	-	-	-	-	-	-	-
e	0	c	e	g	-	-	-	i	-	f	-	-	i	-	-
e	1	-	e	f	-	-	-	-	-	-	-	-	-	-	-
f	#	c	-	f	-	-	-	-	-	-	-	-	i	-	-
f	0	#	0	-	-	0	-	i	-	-	-	-	c	-	-
f	1	-	-	f	-	-	e	-	-	-	g	-	-	-	-
f	g	-	-	-	-	-	-	-	-	-	-	g	h	-	-
f	h	-	-	-	-	-	-	-	-	-	-	g	h	-	-
g	#	#	-	g	1	-	-	-	f	b	f	h	-	-	-
g	0	d	h	g	-	-	#	-	-	0	g	h	-	-	-
g	1	c	h	g	1	-	#	-	i	-	g	h	-	-	-
g	c	d	-	-	-	a	-	-	-	g	-	h	i	-	-
g	d	-	-	-	-	-	-	-	f	e	-	h	-	-	-
g	e	-	-	-	-	-	-	-	-	-	g	-	-	-	-
g	g	-	h	g	g	-	-	-	-	-	g	h	i	-	-
g	h	#	h	g	-	-	i	-	-	-	g	h	i	-	-
g	i	#	-	g	-	-	-	-	g	-	-	-	i	-	-

Table 1: The first half of the transition table of our 14–state universal CA. The upper row shows the state of the considered cell and the first two columns display the states of the left and right neighbor cells. The table-entry "-" means that the combination never occurs during a simulation, thus any of the states could be entered instead of the "-".

		#	0	1	a	b	c	d	e	f	g	h	i	j	q
1	c	c	-	a	a	-	#	-	-	g	1	1	#	-	-
1	d	c	-	d	-	-	-	-	-	d	-	1	i	-	1
1	e	c	-	a	-	-	-	-	-	-	1	-	-	-	-
1	f	c	-	1	-	-	-	-	-	-	g	-	-	-	-
1	g	#	-	1	g	-	i	-	-	q	1	0	#	-	q
1	h	#	0	1	-	h	-	-	-	q	1	0	#	i	j
1	i	#	-	1	g	-	-	1	-	g	1	h	#	-	-
1	j	#	0	1	-	-	-	-	-	-	1	0	-	-	#
1	q	q	-	1	-	-	-	-	-	-	1	0	-	0	q
a	#	#	-	1	1	-	-	-	-	-	e	-	-	-	-
a	0	#	0	-	-	e	#	-	-	-	1	-	-	-	-
a	1	#	-	1	1	-	-	-	-	-	a	-	-	-	-
a	a	-	-	a	-	-	-	-	-	-	a	-	-	-	-
a	b	-	b	-	-	-	-	-	-	-	-	f	-	-	-
a	c	c	-	a	-	-	-	-	-	-	a	-	-	-	-
a	e	-	b	-	-	-	-	-	-	-	-	-	-	-	-
a	f	c	-	-	-	-	-	-	-	-	g	-	c	-	-
a	g	#	-	-	-	-	-	-	-	-	-	-	-	-	-
a	h	-	-	-	-	-	-	-	-	-	-	-	j	-	-
b	#	-	-	-	-	-	#	-	-	-	-	-	-	-	-
b	0	-	0	-	-	0	-	-	-	-	g	b	-	-	-
b	1	-	0	1	1	0	-	-	-	-	a	b	-	-	-
b	a	-	b	a	-	-	-	-	-	-	a	b	-	-	-
b	b	-	b	-	-	-	-	-	-	-	-	b	-	-	-
b	c	-	-	-	-	-	-	-	-	-	-	-	0	-	
b	e	-	b	-	-	-	-	-	-	-	-	-	-	-	-
b	f	-	0	-	-	-	-	-	-	-	-	-	-	-	-
c	#	-	-	d	-	-	-	-	-	-	-	-	-	-	-
c	0	#	0	1	f	0	-	-	-	-	1	b	d	-	-

		#	0	1	a	b	c	d	e	f	g	h	i	j	q
g	j	-	h	g	-	-	-	-	-	-	-	g	-	-	-
g	q	-	h	g	-	-	-	-	-	-	-	g	h	-	1
h	#	i	-	-	-	-	-	-	-	-	-	-	-	#	-
h	0	i	h	-	-	0	-	i	-	-	g	h	#	-	-
h	1	i	h	g	1	0	i	-	-	-	g	h	#	-	-
h	c	i	-	-	-	-	-	-	-	-	-	-	-	-	-
h	g	-	h	g	g	h	-	-	-	-	g	h	-	-	-
h	h	i	h	g	-	h	-	-	-	-	g	h	-	-	-
h	i	i	-	-	-	-	-	-	-	-	-	-	-	-	-
h	j	-	h	g	-	-	-	-	-	-	g	h	-	-	-
h	q	-	h	g	-	-	-	-	-	-	g	h	-	1	j
i	#	-	-	-	-	-	-	h	-	-	-	-	-	-	-
i	0	i	h	c	-	-	0	-	-	1	g	h	i	-	-
i	1	i	-	-	-	-	-	-	-	-	-	-	-	-	-
i	d	-	-	-	-	-	-	-	-	-	-	c	-	-	-
i	g	-	-	-	-	-	j	-	-	-	-	-	-	-	-
i	h	i	h	0	-	h	j	-	-	-	g	h	i	-	-
j	0	-	-	-	-	-	-	-	-	-	-	c	-	-	-
j	1	-	-	-	-	-	-	-	-	-	f	c	-	-	-
j	b	-	-	-	-	-	-	-	-	-	-	c	-	-	-
j	g	-	h	g	-	-	j	-	-	-	-	-	-	-	-
j	h	-	h	g	-	h	j	-	-	-	j	0	-	-	-
j	q	-	-	-	-	-	-	-	-	-	-	-	-	-	q
q	0	-	-	-	-	-	-	-	-	-	g	-	-	-	-
q	1	-	-	1	-	-	-	-	-	-	1	-	-	-	q
q	a	-	-	g	-	-	-	-	-	-	-	-	-	-	-
q	g	-	-	1	-	-	-	f	-	-	d	0	-	#	q
q	h	-	-	-	-	-	-	f	-	-	d	0	-	-	-
q	q	-	-	-	-	-	-	-	-	-	-	-	-	-	q

Table 2: The second half of the transition table of our 14–state universal CA. The upper row shows the state of the considered cell and the first two columns display the states of the left and right neighbor cells. The table-entry "-" means that the combination never occurs during a simulation, thus any of the states could be entered instead of the "-".

References

[1] M.A. Arbib, "Simple Self-Reproducing Universal Automata", *Information and Control* **9** (1966) 177-189.

[2] C. Choffrut and K. Culik II, "On Real-Time Cellular Automata and Trellis Automata", *Acta Informatica* **21** (1984) 393-407.

[3] E.F. Codd, *Cellular Automata*, (ACM Monograph Series, Academic Press, New York, 1968).

[4] K. Culik II and I. Fris, "Topological Transformations as a Tool in the Design of Systolic Networks", *Theoretical Computer Science* **37** (1985) 183-216.

[5] K. Culik II and S. Yu, "Translation of Systolic Algorithms between Systems of Different Topology", Proceedings of the 1985 IEEE and ACM International Conference of Parallel Processings, 1985, 756-763.

[6] K. Culik II and J. Karhumäki, "On the Power of Totalistic Systolic Networks", in preparation.

[7] D. Gordon, "On the Computational Power of totalistic cellular Automata", manuscript.

[8] J.E. Hopcroft and J.D. Ullman, *Introduction to Automata Theory, Languages, and Computation*, (Addison-Wesley, Reading, MA, 1979).

[9] H. Kleine Buening and Th. Ottmann, "Kleine Universale Mehrdimensionale Turingmaschinen, *Elektronische Informationsverarbeitung und Kybernetik* **13** (1977) 179-201.

[10] M. Minsky, *Computation: Finite and Infinite Machines*, (Prentice Hall, Englewood Cliffs, NJ, 1967).

[11] J. von Neumann, *The Theory of Self-Reproducing Automata*, A.W. Burks, ed. (University of Illinois Press, Urbana & London, 1966).

[12] N.H. Packard, "Complexity of Growing Patterns in Cellular Automata", in *Dynamical Behaviour of Automata*, edited by J. Demongeot, E. Goles and M. Tchuente, Proceedings of a workshop held in Marseilles 1983, (Academic Press, New York, 1983).

[13] L. Priese, "Towards a Precise Characterization of the Complexity of Universal and Nonuniversal Turing Machines", *SIAM Journal on Computing* **8** (1979) 508-523.

[14] A.R. Smith III, "Simple Computation-Universal Cellular Spaces", *Journal ACM* **18** (1971) 339-353.

[15] A.R. Smith III, "Real-Time Language Recognition by One-Dimensional cellular Automata", *Journal of Computer and System Sciences* **6** (1972) 233-253.

[16] H. Umeo, K. Morita and K. Sugato, " Deterministic One-Way Simulation of The-Way Real-Time Cellular Automata and its Related Problems", *Information Processing Letters* **14** (1982) 158-161.

[17] S. Wolfram, "Statistical Mechanics of Cellular Automata", *Review of Modern Physics* **55** (1983) 601-644.

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

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

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