

On the maximal cycle and transient lengths of circular cellular automata

Kim Weyns, Bart Demoen

Report CW 375, December 2003



Katholieke Universiteit Leuven
Department of Computer Science
Celestijnenlaan 200A – B-3001 Heverlee (Belgium)

On the maximal cycle and transient lengths of circular cellular automata

Kim Weyns, Bart Demoen

Report CW375, December 2003

Department of Computer Science, K.U.Leuven

Abstract

We prove some conjectures about the maximal cycle and transient lengths of circular cellular automata. These conjectures were left open in the PhD. thesis of Dr. B. Rinaldi. We also prove theorems on the number of circular cellular automata rules that implement these maximal lengths, and characterize completely the set of the possible cycle and transient lengths.

On the maximal cycle and transient lengths of circular cellular automata ^{*}

Kim Weyns [†] Bart Demoen [‡]

5th December 2003

Abstract

We prove some conjectures about the maximal cycle and transient lengths of circular cellular automata. These conjectures were left open in the PhD. thesis [1] of Dr. B. Rinaldi. We also prove theorems on the number of circular cellular automata rules that implement these maximal lengths, and characterize completely the set of the possible cycle and transient lengths.

1 Introduction

The study of cellular automata has recently attracted more attention again thanks to the work published in the controversial book[2] of S. Wolfram. Within this broad field the study of cyclic automata is of particular interest because these cellular automata are very often used to model a great variety of systems. One of the problems studied in this area is the existence of long cycles and transients.

In her PhD. thesis [1], Dr. B. Rinaldi published some conjectures on the maximal possible cycle and transient lengths of circular cellular automata. The proofs of these conjectures in this text are based on the concept of spatial periods and how they influence the behavior of cellular automata. The key construction that makes the proofs easy is a (double) partition of the set of possible strings of a circular cellular automaton. We also provide a good understanding of how long cycles and transients can occur and can even be

^{*}Part of the Master Thesis of the first author - to be published in Dutch

[†]Master Student Computer Science Engineering, K.U.Leuven -
kim.weyns@student.kuleuven.ac.be

[‡]Supervisor

constructed through the use of graphs in this partition. Also we characterize completely the the set of the possible cycle and transient lengths.

We start by introducing the relevant terminology in Section 2. In Section 3 we cite some important theorems from [1] which we need in the proofs later. In Section 4 we define two key functions introduced in [1], and we use these functions to define some final concepts we need in our proofs. In Section 5, we state the original conjectures. The actual proofs are in Sections 6 and 7, together with some further conclusions on cycle an transient lengths. Finally, Section 8 describes some possible extensions to the theorems and future work.

2 Definitions and preliminary lemmas

We begin by developing some notation necessary in the remainder of this document. In this text N always denotes the size of a circular cellular automaton. N is a positive integer.

2.1 Cellular automaton

A one-dimensional cellular automaton of size N with k states, is an array of N square cells, where each cell can be in any of the k states, evolving under a cellular automaton rule. The cells are numbered from 0 to $N - 1$. This defines a natural notion of left and right. Cell 1 is left of cell 2 and right of cell 0, and so on. This array is often represented by a string b of N integers in $0 \dots k - 1$. Such a string b can also be written as $b_0 b_1 \dots b_{N-1}$. The definition of a cellular automaton rule is given in Section 2.5.

Let B be the set of k states. Then the set of strings of length N is denoted by B^N . It is clear that:

$$\#B^N = k^N \tag{1}$$

2.2 Limiting the number of states

From this point in the text on we will limit ourselves to the special case where k , the number of possible states of a cell, is 2. All of the following theorems and formulas can be generalized quite easily to an arbitrary number of states, but this makes the theorems and formulas less clear and harder to understand.

2.3 Circular cellular automaton

In a circular cellular automaton, we imagine the array of cells to be a closed circle. This means we consider cell 0 to be the right neighbor of cell $N - 1$, and of course also cell $N - 1$ to be the left neighbor of cell 0. In this paper we will only discuss circular cellular automata.

2.4 Neighborhood

An r -neighborhood of a cell c in a cellular automaton is a set of r cells of which the position is determined relative to the position of the cell c . In a circular cellular automaton of size N , this relative position is of course to be considered modulo N . For example, the cell 3 cells left of a certain cell, is the same as the cell $N - 3$ cells right of that cell.

2.5 Cellular automaton rule

A cellular automaton rule Φ of size r is a function that maps the current configuration of the r -neighborhood of a cell onto a new state for that cell: $\Phi : \{0, 1\}^r \rightarrow \{0, 1\}$. If we say that a cellular automaton of size N evolves under a cellular automaton rule, it means we repeatedly apply this function onto the neighborhood of all the N cells at the same time, thereby updating every cell depending on the state of all the cells in their neighborhood. This way in every step, a configuration of the cellular automaton is mapped onto another and we can also write $\Phi : B^N \rightarrow B^N$.

When a rule Φ maps a string b onto the string b' , we write this as $\Phi(b) = b'$. When a string b is mapped onto the string b' by applying the rule Φ t times, we write $\Phi^t(b) = \Phi(\Phi(\dots\Phi(b))) = b'$

A cellular automaton rule of size r can be summarized in a rule table. This rule table contains 2^r lines, displaying what the new state of a cell will be for the 2^r possible configurations of the r -neighborhood of that cell.

Figure 1 shows a circular cellular automaton of size 6 evolving under a rule of size 2. The 2-neighborhood here is the cell itself and it's left neighbor. This means the new value of every cell depends only on the value of the old value of the cell itself and it is left neighbor. The starting configuration is represented by the top row of cells, each next step in the evolution is shown under the previous one. The rule is schematically represented by its rule table on the right of the cellular automaton.

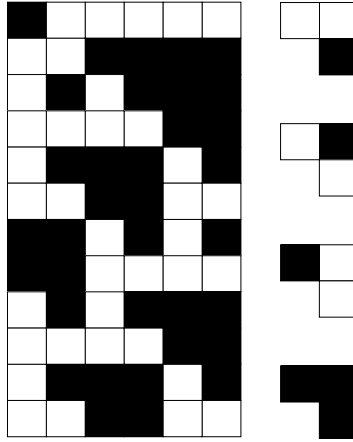


Figure 1: A circular cellular automaton of size 6, with a rule of size 2

2.6 Shift

We call two strings b and b' a shift of each other, if an integer $s > 0$ exist for which $b_i = b'_{(i+s) \bmod N}$ for every i in $0 \dots N - 1$, and we can write this as $\sigma_s(b) = b'$. We say that the evolution of a cellular automaton has reached a shift, if a string is reached that is a shift of a string that was reached before.

Rules in a circular cellular automaton always comply with the following lemma: if rule Φ maps a string b to a string b' , it must also map any shift of b to a similar shift of b' . In symbols this can be written as:

Lemma 1 \forall shift σ_s, \forall cellular automaton rule $\Phi, \forall b \in B^N$

$$\sigma_s(\Phi(b)) = \Phi(\sigma_s(b)) \quad (2)$$

2.7 Spatial period

The spatial period of a string b of length N , represented as $sp(b_0b_1 \dots b_N)$ or $sp(b)$, is defined as the smallest positive integer p for which $b_i = b_{(i+p) \bmod N}$ for all i in $0 \dots N - 1$. It is clear that the spatial period of any string b of length N is a divisor of N . The notion of spatial period is the basis for understanding the structure of the set of strings B^N as we introduce it in Section 4.4.

2.8 Cycles and transients

We already mentioned that the number of possible strings that can be reached in the evolution of a circular cellular automaton is limited by the quantity in equation 1. Since a cellular automaton rule maps a particular string always to the same successor, the evolution of a cellular automaton eventually reaches a cycle, a repeating sequence consisting of a limited number of steps. And we know it reaches this cycle in a finite number of steps. The minimum number of steps between two occurrences of the same string in the evolution of a cellular automaton is called the cycle length of the cellular automaton. The length of the sequence of strings that appears in the beginning of the evolution before the start of the cycle, and that therefore can not appear again in the evolution of this cellular automaton, is called the transient of the cellular automaton. If the starting configuration is already part of the cycle, then we say that the cellular automaton has transient length 0.

Figure 1 shows a cellular automaton where the 3^{rd} line is exactly the same as the 9^{th} line. The first two configurations are never reached again in the evolution of the cellular automaton. So this means this cellular automaton has a transient length of 2, and a cycle length of 6.

3 Important theorems from [1]

In her thesis [1], Billie Rinaldi proves the following important theorems which we need in the remainder of this document.

Theorem 1 *Given strings $b, b' \in B_N^2$, the string b' is the result of applying some cellular automaton rule to b , if and only if the spatial period of b' is a positive integer divisor of the spatial period of b .*

Theorem 2 *Given a cellular automaton rule ϕ , strings b and $b' \in B_N^2$, and an integer $m > 1$ such that a shift appears in the first $m - 1$ steps in the evolution of b under ϕ , there exists a cellular automaton rule ψ such that $\psi^t(b) = \phi^t(b)$ for $t = 0 \dots m - 1$ and $\psi^m(b) = b'$ if and only if $b' = \phi^m(b)$.*

This theorem says that if the evolution of two cellular automata starting from the string b , is the same up to a shift, their future evolution must also be the same. This means that as soon as a shift occurs in the evolution of a cellular automaton, the further evolution of the cellular automaton is fixed.

Theorem 3 *Given a cellular automaton rule ϕ , strings b and $b' \in B_N^2$, and an integer $m > 1$ such that a shift has NOT appeared in the first $m - 1$ iterations of b under ϕ , there exists a cellular automaton ψ such that $\psi^t(b) = \phi^t(b)$ for $t = 0 \dots m - 1$ and $\psi^m(b) = b'$ if and only if the spatial period of $\phi^{m-1}(b)$ is a positive integer multiple of the spatial period of b' .*

4 The functions $P(N)$ and $T(N)$

In [1], Billie Rinaldi also defines the following two important functions P and T .

4.1 $P(N)$

$$P(N) = 2^n - \sum_{d \in D_N \setminus \{N\}} P(d).$$

where D_N is the set containing all the divisors of N .

4.2 $T(N)$

$$\begin{aligned} T(N) &= 1 && \text{for } N = 1 \\ &= \frac{P(N)}{N} + T(n_1) && \text{for } N > 1. \end{aligned}$$

where $n_1 = \max D_N \setminus \{N\}$.

Table 1 shows the first 7 values of the functions P and T .

N	$P(N)$	$T(N)$
1	2	1
2	2	2
3	6	3
4	12	5
5	30	7
6	54	12
7	126	19

Table 1: Some values of P and T

4.3 Some properties of the functions P and T

In [1], Billie Rinaldi proves that the $P(p)$ as defined above, is the number of strings of a given length N which have spatial period p , as long as p is a divisor of N . In symbols this gives

$$\forall p \in D_N : \#\{b \in B^N | sp(b) = p\} = P(p) \quad (3)$$

In the remainder of this document we will also need the following properties of the functions $P(N)$ and $T(N)$ which can easily be checked for small N , and proven for larger N , but the proofs are omitted here. These properties are

Lemma 2 $P(N)$ is strictly increasing with N for $N > 2$

Lemma 3 $T(N)$ is strictly increasing with N for $N > 0$

4.4 Partitioning the set B^N

Our proofs rely on a particular partitioning of the set of strings B^N . We can divide the set of strings B^N in disjunct subsets by spatial period of the strings. We refer to the subset of strings with spatial period d_i as $V_{d_i}^N$.

For example:

$$\begin{aligned} V_1^4 &= \{1111, 0000\} \\ V_2^4 &= \{1010, 0101\} \\ V_4^4 &= \{1000, 0100, 0010, 0001, \\ &\quad 1100, 0110, 0011, 1001, \\ &\quad 1110, 0111, 1011, 1101\} \end{aligned}$$

If a string b has a spatial period p , then of course every shift of b also has spatial period p . For a string b with spatial period p , p different strings can be found which are a shift of b , including b itself.

Now we can again divide the sets $V_{d_i}^N$ in $\frac{P(d_i)}{d_i}$ subsets where every subset contains just those strings which are a shift of each other. Such a partition is represented in the Figure 2 for $N = 6$, where every element of B^6 is represented by a black dot, the subsets $V_{d_i}^6$ are represented by dotted lines and their subsets by thin, full lines.

When we represent the set B^N as in figure 2, we can represent the effect of a cellular automaton rule Φ on the set B^N as a directed graph, where the node representing a string b is connected to the node representing the

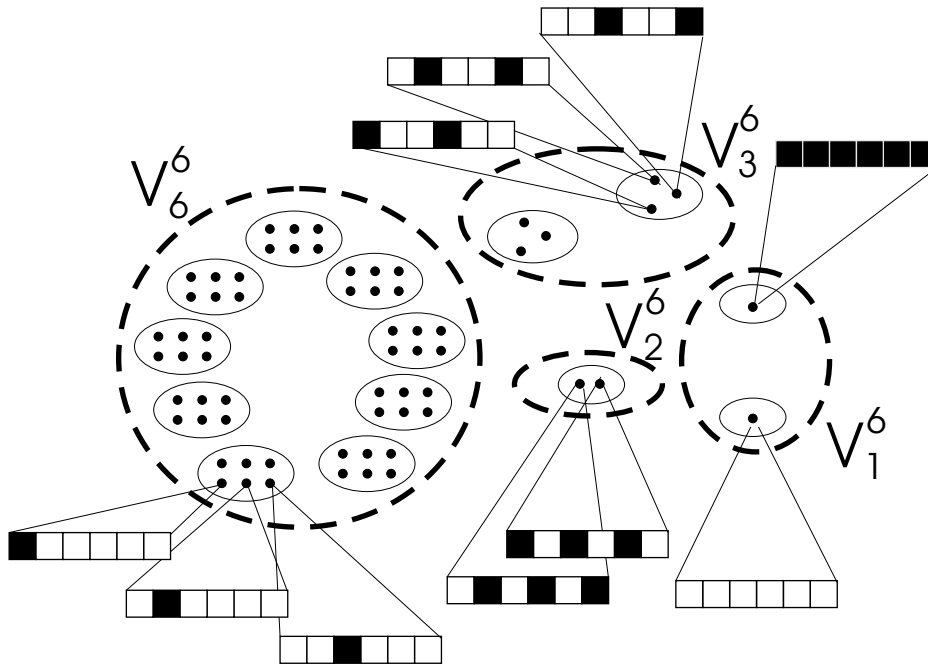


Figure 2: The set V^6 , containing 64 strings

string's successor $\Phi(b)$. There are some constraints on this graph. First of all, all nodes must have an out-degree of exactly 1. Also the graph must satisfy lemma 1. And because of theorem 1, no edges can go from a subset with spatial period p_1 to a subset with spatial period p_2 unless p_2 is a divisor of p_1 . An example of such a graph can be seen in figure 3. Under the graph you can see the evolution caused by this rule when starting from different starting strings.

In such a graph it becomes very easy to find the possible cycle and transients created by a certain cellular automaton rule. And a good understanding of this graph can also help us construct a rule with a maximal cycle or transient.

Note that the partition of B^N is based on the equivalence relations *has the same spatial period as* and *is a shift of*.

5 The conjectures

In [1], Billie Rinaldi left the two following conjectures unproven.

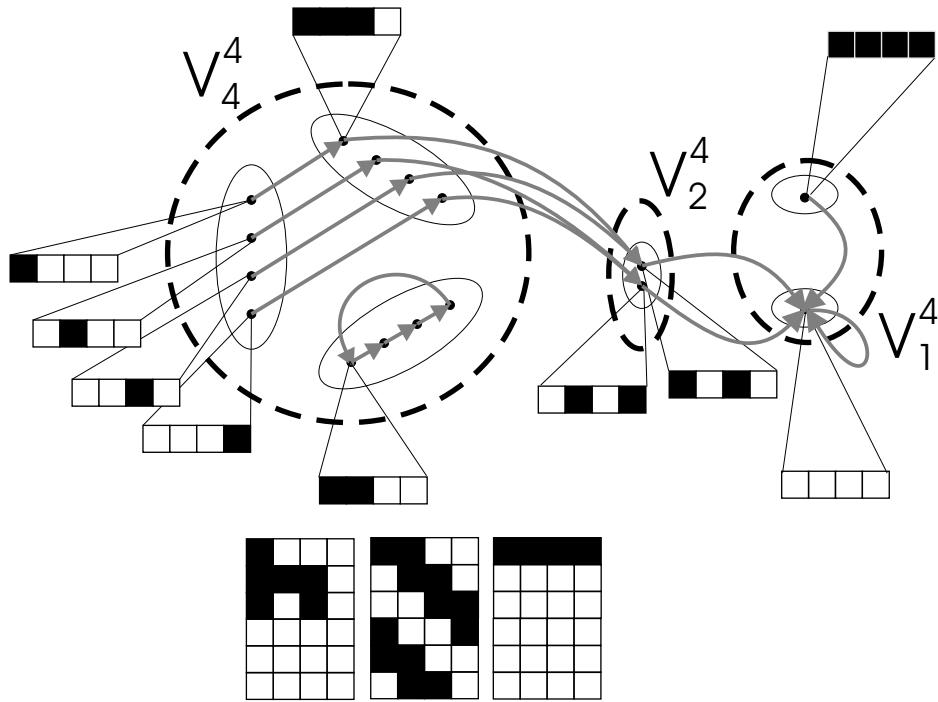


Figure 3: The graph of a cellular automaton rule

Conjecture 1 *The maximal cycle length of a circular cellular automaton of size N is $P(N)$, and this cycle length can be achieved through some cellular automaton rule of size N .*

Conjecture 2 *The longest possible transient before a cycle in the evolution of a cellular automaton of size N is $T(N)$, and this transient length can be achieved through some cellular automaton rule of size N .*

Our main aim is to propose a proof for these two conjectures in the remainder of this document.

6 Maximal Cycle Length

We are now ready to prove Conjecture 1 on the maximal cycle length for a given size N . The proof consists of two parts. First we prove an upper

bound for the cycle length and then we construct a cellular automaton which realizes this upper bound.

6.1 Part I: no cycle is longer than $P(N)$

To prove that no cycle can exist longer than $P(N)$, it is sufficient to point out that as a consequence of Theorem 1 all strings in a cycle must have the same spatial period. The highest possible spatial period of a string of length N is N . From Lemma 2 it is now clear that the maximum number of strings which could be in a cycle is indeed $P(N)$.

6.2 Part II: a cycle with length $P(N)$ exists

Instead of merely proving the existence, we present a constructive proof, i.e. a method to construct a cellular automaton which produces a cycle length of $P(N)$. The way a cellular automaton can produce a maximal cycle can also be seen in the graph in figure 4 for $N = 4$, where the edges that are not part of the maximal cycle have been left out.

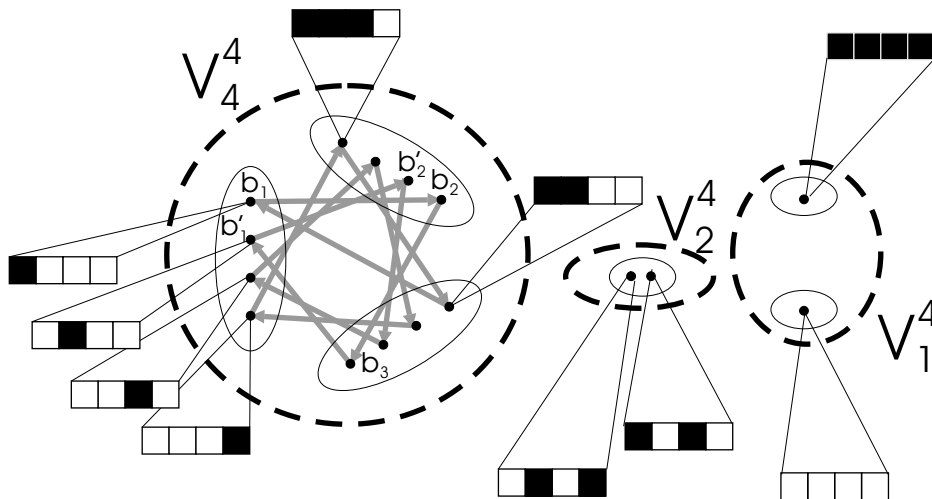


Figure 4: A maximal cycle of size 4

As starting configuration for the cellular automaton we take an arbitrary string $b_1 \in V_N^N$. To construct the rule table of a rule that accomplishes a cycle length of $P(N)$ we can go through the following steps:

First choose a second string b_2 , also from V_N^N , but not a shift of b_1 . Theorem 1 says that we can now always find a cellular automaton rule ϕ_1 which maps b_1 to b_2 .

Then choose a third string b_3 , again from V_N^N , but not a shift of b_1 or b_2 . According to Theorem 3 a cellular automaton rule ϕ_2 exists which maps b_1 to b_2 and b_2 to b_3 .

Now repeat this procedure for all the remaining $\frac{P(N)}{N}$ subsets. Then we have a cellular automaton rule $\phi_{\frac{P(N)}{N}-1}$ mapping b_i to b_{i+1} for $i = 1 \dots \frac{P(N)}{N} - 1$, with all the b_i 's not shifts of each other.

As a last step, take another string b'_1 that is the string b_1 shifted by 1 (either left or right). Since no shift has been encountered so far in the evolution of $\phi_{\frac{P(N)}{N}-1}$, Theorem 3 tells us that we can still find a cellular automaton rule $\phi_{\frac{P(N)}{N}}$ which not only maps all the previously described strings the same way as $\phi_{\frac{P(N)}{N}-1}$, but also map $b_{\frac{P(N)}{N}}$ to the string b'_1 .

Now this evolution cellular automaton is completely determined according to Theorem 2 and Lemma 1. The cellular automaton cycles through the $\frac{P(N)}{N}$ subsets, shifting one position in every cycle. Therefore the number of steps before reaching the string b_1 again, equals $N * \frac{P(N)}{N} = P(N)$.

A possible rule table of this cellular automaton rule can very easily be composed now. Since the size of the rule is N , the state of a cell can depend on the states of all the cells in the previous step. For example, consider the case where we choose to take the rule table where the new state of a cell depends on the previous state of the cell itself and the state of the $N - 1$ cells right of this cell. For each of the first $P(N)$ lines in the evolution of this cellular automaton, there is a line in the rule table, saying what the most left cell will be in the next step. To complete the rule table, we can fill in the remaining $2^N - P(N)$ lines with an arbitrary result, because these configurations do not appear in the evolution of the cellular automaton since they have a spatial period lower than N .

■

From this proof we can see that any string $b \in V_N^N$ is reached in this maximal cycle. And this means that to test if a rule can produce a maximal cycle, we can start the rule on an arbitrary string $b \in V_N^N$.

In figure 6.2 you can see some maximal cycles for cellular automata of sizes 1 to 5.

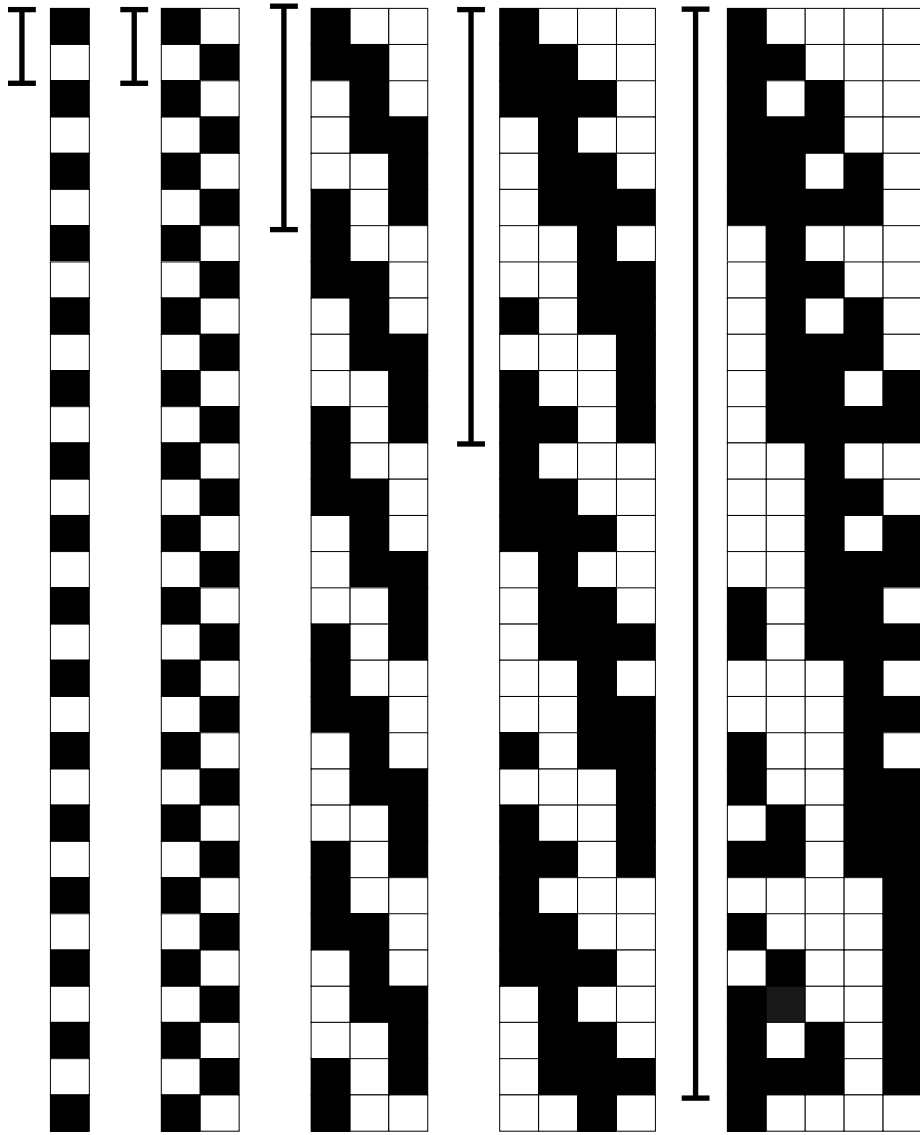


Figure 5: Maximal Cycles for $N = 1$ to 5

6.3 Occurrence of maximal cycles

We are now ready to study the following question: for a given N , how many cellular automata rules exist which realize this maximal cycle length?

Take as an example $N = 7$. Since the new state of a cell depends on the states of all the cells in the previous step, there is no choice about which 7-neighborhood is used in the rule. This means the total number of rules for $N = 7$ is 2^{2^7} or $\approx 3.4e38$. We also know $P(7) = 2^7 - 2 = 126$ and $\frac{P(7)}{7} = 18$.

Before answering the question how many cellular automata exist with the maximal cycle, we answer the question how many different maximal cycles there are. Since any element of V_7^7 is in every maximal cycle, we can choose one arbitrary string from V_7^7 to always be the start configuration of the cycle.

Maximal cycles can then only differ in the order in which they visit the remaining different subsets of V_7^7 , in the first chosen representative of each subset, and in the shift they accomplish after 18 steps: from then on the cycle is fixed.

We can choose between $(18 - 1)!$ possibilities for the order of the subsets. For each of those subsets we can also choose which of the seven strings is the first string encountered. Also we can choose the size of the first shift encountered, between 1 and 6¹. In total, this means that there are $6 \cdot 7^{17} \cdot (17)! \approx 5e29$ possible maximal cycles, where each of them can be generated by a different cellular automaton rule. But now we still have not taken into account that more than one cellular automaton rule can generate the same cycle. There are 2 strings of size 7 which do not have spatial period 7, and where the behavior of the rule does not matter for the maximal cycle. This means that in total there are approximately $2e30$ rules which give a cycle of maximal length when started on any string with spatial period 7. This number might seem very large, but it is still only one rule out of every 170 million possible rules.

It is obvious that the number of maximal rules increases with N , but the total number of rules increases even faster, so the fraction of maximal rules becomes smaller. A similar way of reasoning yields the results in Table 2.

With a similar reasoning we can also write a general formula for this number of maximal rules. Because we can choose the order of the subgroups in the maximal cycle, this number of maximal rules contains the factor $\left(\frac{P(N)}{N} - 1\right)!$. Because for every subgroup we can choose between N strings for the first representative of the subgroup, there is also a factor $N^{\frac{P(N)}{N} - 1}$.

¹every choice leads to a maximal cycle, because 7 is prime

N	Number of Rules	Number of Maximal Rules
1	4	1
2	16	4
3	256	24
4	65536	1024
5	4.29e9	6e6
6	1.84e19	1.39e14
7	3.4e38	1.99e30

Table 2: The number of rules that can produce a maximal cycle

For the first shift achieved, we can also choose the size. For a prime number we can take any shift between 1 and $N - 1$. For a non-prime number we must make sure to choose a shift s that does not share any real divisors with N or the cellular automaton does not reach the maximal cycle length, i.e. $\gcd(s, N) = 1$. Finally, to reach the number of maximal rules, we still have to multiply the number of possible maximal cycles, with 2 to the power of the number of strings of size N that do not appear in a maximal cycle because they do not have spatial period N . For a prime number this number is always 2, but for a non-prime it is not constant and larger than 2. For a prime number we find the formula

$$\# \text{ maximal rules} = 4 \cdot (N - 1) \cdot \left(\left(\frac{P(N)}{N} - 1 \right) ! \right) \cdot N^{\frac{P(N)}{N} - 1}. \quad (4)$$

For a non prime number, the formula becomes more complicated and that would lead us too far for this paper.

6.4 Possible Cycle Lengths

Now we know the maximal cycle length, we can also raise the question if all possible cycle lengths between 1 and $P(N)$ can be found if we just choose the right rule and starting string.

To answer this, look back to figure 4. We already mentioned that cycles must always stay inside the same subset V_d^N .

Let us first consider the subsets with spatial period $p < N$. All strings in the subset with a spatial period p are just strings of length p repeated to form length N . Since we consider cyclic boundary conditions, they of course behave exactly like the cellular automata of size p . Therefore we can conclude that all cycle lengths which can be found in cellular automata of

a size which is a divisor of N , can also be found in some cellular automata of size N .

Now consider the subset V_N^N . We already said this subset consists of $\frac{P(N)}{N}$ subsets of size N . Not all subsets have to be involved in a cycle. The number of subsets involved can be any number between 1 and $\frac{P(N)}{N}$. It can also be observed that a cycle in a circular cellular automaton must satisfy the following constraint concerning the subsets that participate in the cycle: such a cycle always visits every subset an equal number of times, and this number of times can be any divisor of N , because it is the order of a subgroup of a cyclic group. Together, this means that any cycle length that can appear must be the result of multiplying a number in $\left[1, \frac{P(N)}{N}\right]$ with a divisor of N . And since there are no more constraints for a cycle, for any such number, cycles of this length exist and can be constructed.

Before we write this down in a formula, we note that all cycle lengths which can be achieved in a subsets with spatial period $p < N$, can also be achieved in the subset with spatial period N . The proof for this relies on the fact that any divisor of a divisor of N , is of course also a divisor of N , and the fact that the function $\frac{P(N)}{N}$ increases in N when $N \geq 3$. This can be checked easily for small N , and be proven quite easily for large N , since $P(N) = O(2^N)$. So now we can summarize this in the following theorem.

Theorem 4 *A cyclic cellular automaton of size N must always have a cycle length in the following set*

$$C_N = \{p \in [1, P(N)] \mid \exists k \in \left[1, \frac{P(N)}{N}\right], \exists l \in D_N : p = k * l\} \quad (5)$$

And for all numbers in this set, cellular automata can be constructed that have exactly this cycle length.

For example, in a circular cellular automaton of size 4 the only possible periods are: $C_4 = \{1, 2, 3, 4, 6, 8, 12\}$.

7 Maximal transient length

For the proof of the maximal transient length we follow a similar course of reasoning as in the proof of the maximal cycle length. First we show that $T(N)$ is an upper bound for the maximal transient length, and then we construct a cellular automaton that has a transient length of exactly $T(N)$.

Unlike the maximal cycle proof, the maximal transient proof is based on induction, just as $T(N)$ is defined by induction. The base case has $N = 1$.

For a cellular automaton of size $N = 1$, there are only two possible strings, and it is clear that the only possible transients in this case are of length 0 or 1. For an example of the transient of length 1, see the most left cellular automaton in figure 7.2.

7.1 Part I: no transient is longer than $T(N)$

Not all strings in a transient need to have the same spatial period, so in a transient in a cellular automaton of size N , we can find both strings with spatial period N and with a spatial period $p < N$. Theorem 1 must of course hold for every step in the transient, so the strings with spatial period N must always come before the other strings.

Since we know that no shifts can appear in a transient, there can be a maximum of $\frac{P(N)}{N}$ strings with spatial period N participating in the transient. After the first string with spatial period $p < N$ has been reached, only strings can appear that have as spatial period that is a divisor of p , and the cellular automaton behaves just like a cellular automaton of size p . The maximum number of steps before reaching a cycle in a cellular automaton of size p is $T(p)$ because of the induction. Theorem 3 tells us this number is largest for the largest possible value for p , and this is the largest divisor of N . From this we can now conclude that the longest possible transient in a cellular automaton is indeed $T(N)$ as defined above.

7.2 Part II: a transient of length $T(N)$ exists

Instead of merely proving the existence, we present a constructive proof, i.e. a method to construct a cellular automaton which produces a cycle length of $T(N)$. The way a cellular automaton can produce a maximal transient can also be seen in figure 7.2 for $N = 6$, also here the edges that are not part of the maximal transient have been left out.

The first steps in the construction of this cellular automaton are the same as for constructing a maximal cycle. As starting configuration for the cellular automaton we take an arbitrary string $b_1 \in V_N^N$. To construct the rule table of a rule that accomplishes a cycle length of $P(N)$ we can go through the following steps:

First choose a second string b_2 , also from V_N^N , but not a shift of b_1 . Theorem 1 says that we can now always find a cellular automaton rule ϕ_1 which maps b_1 to b_2 .

Then choose a third string b_3 , again from V_N^N , but not a shift of b_1 or b_2 . According to Theorem 3 a cellular automaton rule ϕ_2 exists which maps

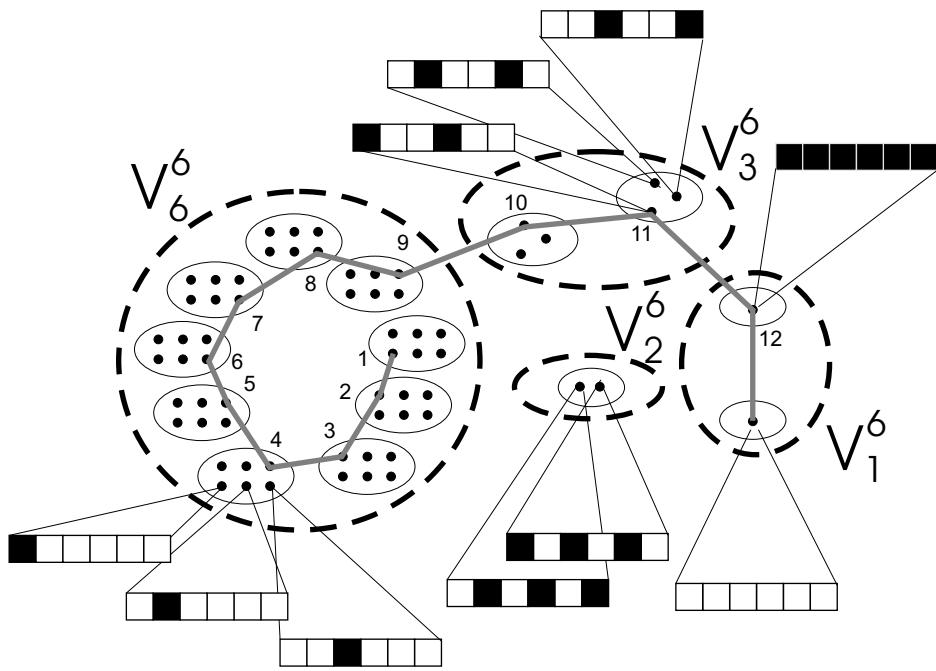


Figure 6: How a maximal transient for $N = 6$ can exist

b_1 to b_2 and b_2 to b_3 .

Now repeat this procedure for all the remaining $\frac{P(N)}{N}$ subsets. Then we have a cellular automaton rule $\phi_{\frac{P(N)}{N}-1}$ mapping b_i to b_{i+1} for $i = 1 \dots \frac{P(N)}{N} - 1$, with all the b_i 's not shifts of each other.

As a next step, take an arbitrary string b'_1 with spatial period $p < N$, the largest divisor of N . Since no shift has been encountered so far in the evolution of $\phi_{\frac{P(N)}{N}-1}$, Theorem 3 tells us that we can still find a cellular automaton rule $\phi_{\frac{P(N)}{N}}$ which not only maps all the previously described strings the same way as $\phi_{\frac{P(N)}{N}-1}$, but also map $b_{\frac{P(N)}{N}}$ to the string b'_1 .

Because of the induction we know that we were already able to construct a cellular automaton with transient $T(p)$, starting from a string consisting of the p first cells of the string b'_1 . We can now continue the construction of our cellular automaton in exactly the same way as is we are building the automaton with transient $T(p)$. But instead of using strings of length p , we use strings of length p repeated $\frac{N}{p}$ times to form length N . This is of course only possible, because we know that none of the strings with spatial period N we used before can be a shift of a repeated string with spatial period p . Then we have constructed a cellular automaton of size N with transient $\frac{P(N)}{N} + T(p) = T(N)$.

A possible rule table of this cellular automaton rule can very easily be composed now. Since the size of the rule is N , the state of a cell can depend on the states of all the cells in the previous step. For example, consider the case where we choose to take the rule table where the new state of a cell depends on the previous state of the cell itself and the previous state of the $N-1$ cells right of this cell. For all the strings with a spatial period that takes part in the maximal transient, we can easily deduct the corresponding line in the rule table from the evolution of the cellular automaton. To complete the rule table, we can fill in the remaining lines (with spatial periods that did not appear in this cellular automaton) with an arbitrary result, because these lines do not influence the evolution of the cellular automaton. ■

From this proof we can see that such a maximal transient always ends with strings with spatial period 1, so in a cycle of only black cells or only white cells.

In figure 7.2 you can see some maximal transients for cellular automata of sizes 1 to 6, that all end in a cycle of only white cells.

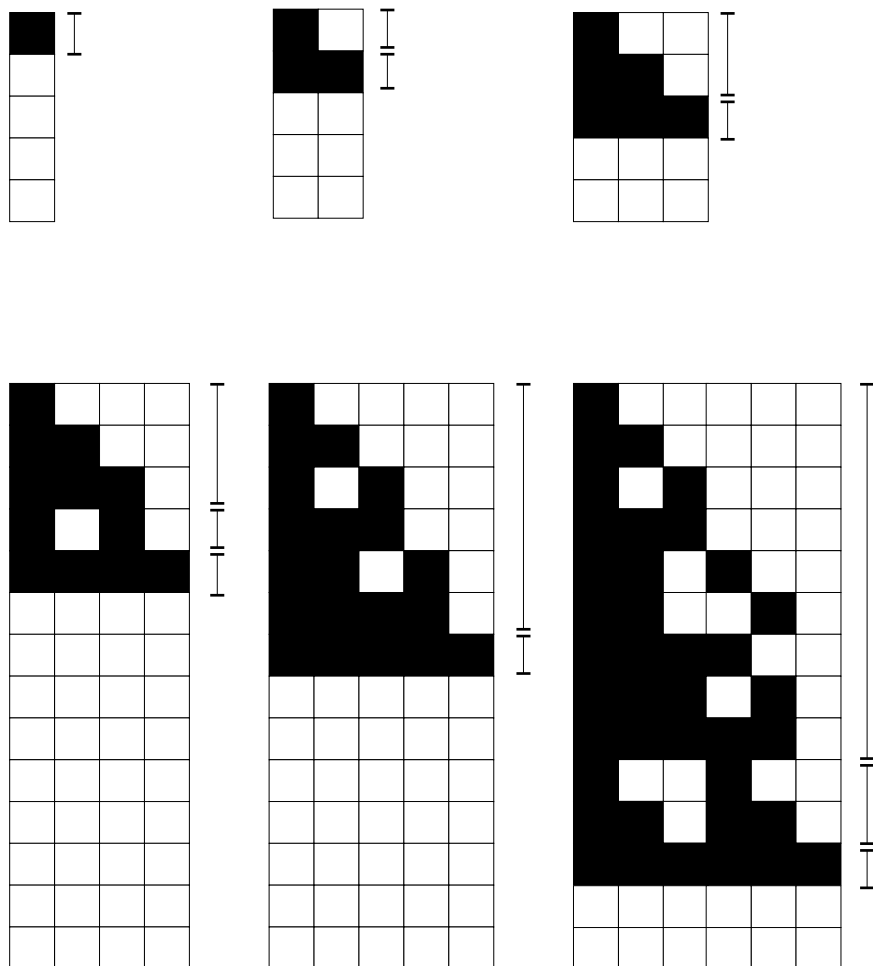


Figure 7: Maximal Transients for $N = 1$ to 6

7.3 Possible Transient Lengths

This is easier than finding all possible cycle lengths. It can be easily understood that all transient lengths from 0 to $T(N)$ are possible. If we want to construct a cellular automaton rule that has a certain transient length l , we can first follow the procedure for constructing a maximal transient for l steps. Since in this transient there is no shift, theorem 3 tells us that we can now find a cellular automaton that has an evolution for l steps just like the maximal transient, and then maps the l^{th} string to all white cells. Since still no shift has occurred, according to theorem 3 we can now also find a cellular automaton that does all of the above, and also maps all white cells onto all white cells. Now this cellular automaton has reached a cycle, and we know it has transient length l . Therefore conjecture 2 can be generalized to the following theorem.

Theorem 5 *The length of the transient of a cellular automaton of size N must be in the set $\{0, 1, \dots, T(N)\}$, and for each number in this set, a cellular automaton of size N with this transient length exists.*

8 Conclusions and future work

We have been able to prove the theorems stated in section 5, and also we have been able to expand these theorems to include all possible cycle and transient lengths. We have also provided constructive methods for finding cellular automata that produce any possible cycle or transient length.

The context in which the theorems for maximal cycle and transients were proven, was circular cellular automata. We will work on similar theorems for cellular automata with a fixed boundary.

Another interesting question that could be asked is which cycle and transient lengths can be achieved when we only allow rules with a limited size r on a cellular automaton of size N . Preliminary research shows that this is a very difficult topic, where much work is still to be done. Also, progress in this field would have a much higher practical value because it would allow us to expand the size of the cellular automaton, without having to deal with very complicated rules.

Some of the theorems stated in this document can be rephrased directly as graph problems ². Also there is clearly a link with the study of commutative rings. If time permits, we will investigate whether a more theoretical

²A similar link between graph theory and cellular automata is discussed more in [3]

and thorough exploration of these links with graph theory and group theory can lead to new results about cellular automata.

References

- [1] Billie J. Rinaldi, A Cellular Automaton Inverse Problem. PhD. Thesis, Rensselaer Polytechnic Institute, Troy, New York, August 2003
- [2] Stephen Wolfram, A New Kind of Science. Wolfram Media, Inc., 2002
- [3] Harold V. McIntosh, Linear Cellular Automata. Universidad Autonoma de Puebla, Puebla, Mexico, 1990
<http://delta.cs.cinvestav.mx/~mcintosh/oldweb/lcau/lcau.html>