Parity Filter Automata

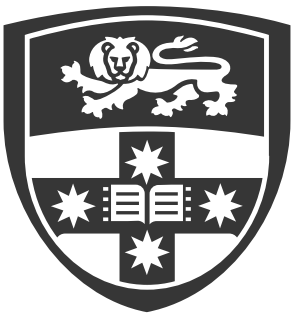
Alexander Tan

Supervisor: Professor Nalini Joshi

An essay submitted in partial fulfillment of
the requirements for the
Science Summer Research Program

School of Mathematics and Statistics

University of Sydney



February 2021

Contents

Introduction

A cellular automaton is a discrete mathematical system in which simple rules often lead to complex emergent behaviour. A typical cellular automaton consists of cells arranged in a line or grid, each of which has a value chosen from a finite set. At each time step, the value of every cell is updated simultaneously based on an update rule involving the values of the cell and its neighbouring cells at the previous time iteration.

Cellular automata are often used to model complex physical phenomena, such as those arising in physics, chemistry and biology. For instance, cellular automata were used to model the large-scale spatial patterns of naturally occurring mussel beds [14], to model HIV infections [5], and to model reaction-diffusion equations [4]. Cellular automata have also been used as random number generators [12], and to create cryptographic ciphers [11].

In Chapter 1 we provide a brief introduction to different types of cellular automata. A particular class of cellular automata known as parity filter automata are introduced in Chapter 2, and some existing results about parity filter automata are summarised in Chapter 3. In Chapter 4 we present original results regarding the behaviour of certain initial configurations under a class of generalized parity filter automata.

CHAPTER 1

Overview of cellular automata

In this chapter we provide a brief overview of different kinds of cellular automata.

1.1. Elementary cellular automata

Elementary cellular automata are one of the simplest examples of cellular automata, and have been studied extensively in [13]. Here, each cell takes on value 0 or 1, and the cells are arranged in a one dimensional line. The update function of a cell is dependent on the cell itself and its two immediate neighbours to the left and right.

More formally, if we denote the value of a cell at position i and time t as $a_i^t \in \{0, 1\}$, then the value of the cell at the next time iteration $t + 1$ is given by

$$a_i^{t+1} = \phi(a_{i-1}^t, a_i^t, a_{i+1}^t) \tag{1.1}$$

for some function $\phi : \{0, 1\}^3 \longrightarrow \{0, 1\}$ with the requirement that $\phi(0, 0, 0) = 0$.

Example 1.2. Consider the function ϕ given in Table 1.1.

$a_{i-1}^t, a_i^t, a_{i+1}^t$	$1,1,1$	$1,1,0$	$1,0,1$	$1,0,0$	$0,1,1$	$0,1,0$	$0,0,1$	$0,0,0$
a_i^{t+1}	0	0	0	1	1	1	1	0

TABLE 1.1. An update function ϕ with canonical code 30.

Successive applications of this rule to a simple initial configuration is shown in Figure 1.1. White squares denote cells with value 0, and black cells denote cells with value 1. Time flows downwards, so that the top row represents the initial configuration. In this example, the initial configuration consists of a single cell with value 1, with the rest of the cells taking value 0.

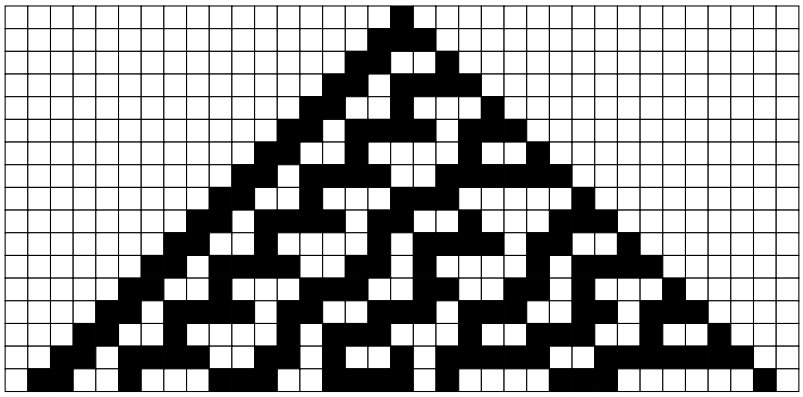


FIGURE 1.1. Rule 30.

The function ϕ is described fully by the bottom row of Table 1.1. For this particular example, reading off the bottom row yields the string 00011110, which, when read in binary, is the decimal number 30. This number is called the canonical code or rule, and elementary cellular automata are often referred to by their canonical code instead of an explicit update function ϕ.

In general there are 256 possible canonical codes identifying different elementary cellular automata rules. This can be observed by noting there are two possible values for each entry in the bottom row of Table 1.1, and $2^3 = 8$ entries, giving a total of $2^8 = 256$ unique tables.

What is remarkable is that despite the simplicity of the rules locally, complex emergent behaviour often arises. Rules 30 and 45 can be used to create random number generators [12], and rule 110 has been proven to be Turing complete [3].

1.1.1. Neighbourhood extension. Elementary cellular automata can easily be extended to involve more complex rules. For example, we may choose to have the update function ϕ depend on a larger neighbourhood of radius r surrounding the cell, i.e.

$$a_i^{t+1} = \phi(a_{i-r}^t, a_{i-r-1}^t, \cdots, a_i^t, \cdots, a_{i+r}^t), \tag{1.3}$$

again with the requirement that $\phi(0, \cdots, 0) = 0$. Here, the update neighbourhood is a strip of $2r + 1$ cells centered around a_i^t.

Note that even with $r = 2$, there are already $2^{32} \approx 4.2 \times 10^{10}$ possible rules. As it can be infeasible to study such a large number of rules, interesting subsets of these rules are often studied instead. One such subset are the class of "totalistic" rules [10] in which the value of a cell depends on the sum of its neighbouring cells. For the case $r = 2$, this sum is an integer ranging from 0 to 10. The update function is then a function

$\phi : \{0, 1, 2, \cdots, 10\} \longrightarrow \{0, 1\}$. There are $2^{11} = 2048$ different possible totalistic rules.

Example 1.4. An example of an update function ϕ based on a totalistic rule for $r = 2$ is given in Table 1.2.

$a_{i-1}^t + a_i^t + a_{i+1}^t$	10	9	8	7	6	5	4	3	2	1	0
a_i^{t+1}	0	0	0	0	1	0	1	0	0	1	0

TABLE 1.2. The update function corresponding to rule 82 in the class of totalistic cellular automata with update radius $r = 2$.

We can again devise a canonical code for this class of automata by reading from the bottom row of Table 1.2 and interpreting it as binary, which in this case produces the decimal number 82.

Iterations of rule 82 applied to a simple initial configuration are shown in Figure 1.2.

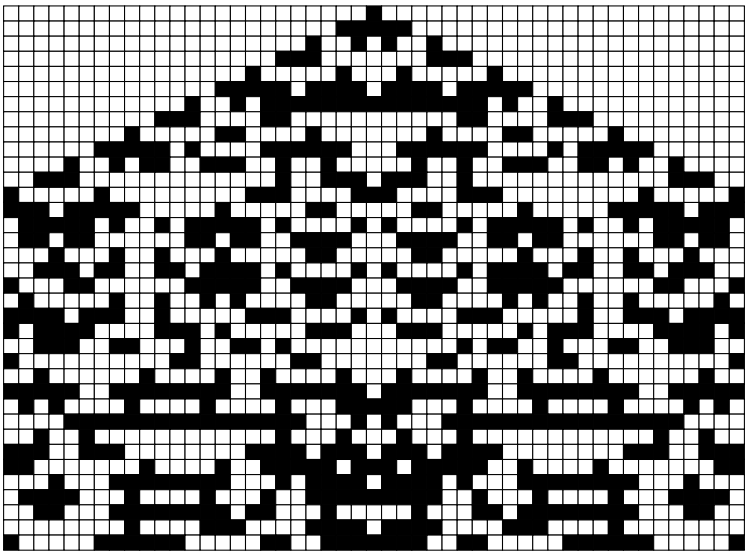


FIGURE 1.2. Rule 82 in the class of totalistic cellular automata with update radius $r = 2$.

1.1.2. Configuration extension. We may also choose to increase the number of states for each cell, say $a_i^t \in \{0, 1, \cdots, k-1\}$. Again, with this extension, the number of rules grows very large, so we study totalistic rules instead.

Example 1.5. A simple example is to consider rules $k = 3$ states. Of course we could also extend the neighbourhood radius as well, but for the sake of example, let us consider the simple case of having the radius $r = 1$. As each cell has 3 neighbours each having value 0, 1, or 2, the total neighbour sum is an integer between 0 and 6. The update function ϕ is then a function of the form $\phi : \{0, 1, 2, 3, 4, 5, 6\} \longrightarrow \{0, 1, 2\}$. An example of such a function ϕ is given in Table 1.3.

$a_{i-1}^t + a_i^t + a_{i+1}^t$	6	5	4	3	2	1	0
a_i^{t+1}	0	2	1	2	1	1	0

TABLE 1.3. The update function corresponding to rule 633 in the class of totalistic cellular automata with $k = 3$ states.

To assign a canonical code, we read off the bottom row in Table 1.3 and interpret it in base 3, which in this case results in the decimal number 633.

Iterations of rule 633 applied to a simple initial configuration are shown in Figure 1.3. To display three states visually, we use white to denote cells with value 0, grey for cells with value 1, and black for cells with value 2.

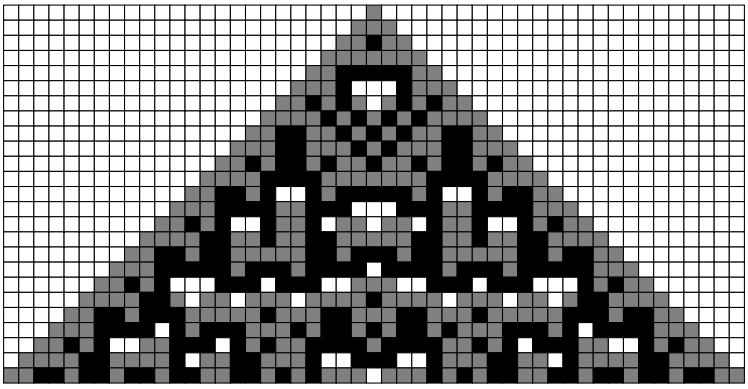
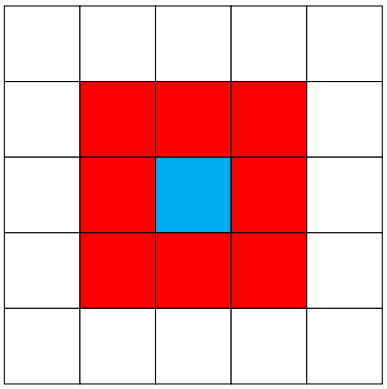


FIGURE 1.3. Rule 633 in the class of totalistic cellular automaton with $k = 3$ states.

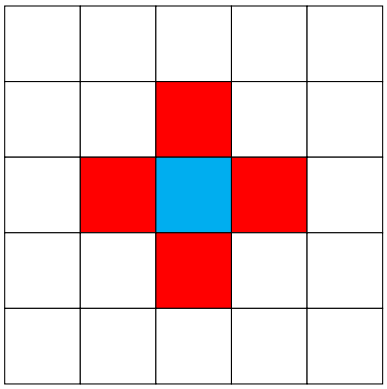
1.2. Two dimensional cellular automata

All the ideas about one-dimensional cellular automata can be extended to higher dimensions. For the sake of visualization, in this section we study the two dimensional case. A natural generalization into 2D is for cells to be laid out in a square grid, as opposed to a line.

In a square grid, there are two common ways of defining a cell's neighbourhood: the 4-neighbourhood (Von Neumann neighbourhood) or the 8-neighbourhood (Moore neighbourhood), as illustrated in Figure 1.4.



(A) A cell (blue) and its 8-neighbourhood (red).



(B) A cell (blue) and its 4-neighbourhood (red).

FIGURE 1.4

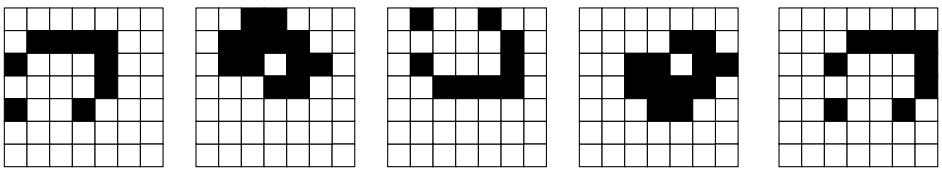
Again, due to the large number of possible rules in 2D cellular automata, totalistic rules are often studied instead.

1.2.1. Game of Life. One of the best known 2D cellular automata is Conway's Game of Life [2]. In Life, each cell has one of two configurations, typically referred to as "dead" or "alive". The 8-neighbourhood is used, and cells are updated to the following totalistic rule.

(1) An alive cell with exactly two or three alive neighbours stays alive in the next iteration. Otherwise it dies as if by underpopulation or overpopulation.
(2) A dead cell with exactly three alive neighbours becomes alive in the next iteration, as if by reproduction. Otherwise it remains dead.

Example 1.6. Figure 1.5 shows the evolution of a pattern called a "lightweight spaceship". Black cells represent "alive" cells and white cells represent "dead" cells. All cells outside the diagram are assumed to be "dead".

The lightweight spaceship displays periodic behaviour in the sense that after 4 iterations, the original pattern reappears 2 units to the right.



(A) $t = 0$. (B) $t = 1$. (C) $t = 2$. (D) $t = 3$. (E) $t = 4$.

FIGURE 1.5. Four iterations of the "lightweight spaceship" pattern in Life.

1.2.2. Other tessellations. Two-dimensional cellular automata do not necessarily have to be aligned in a square grid. Cellular automata on other tilings such as triangular and hexgonal tilings, or even asymmetrical tilings such as pentagonal tilings can be studied. For example, totalistic rules similar to Life have been studied on triangular, pentagonal, and hexagonal tilings in [1].

The notion of neighbourhood needs to be extended to these tilings. As an example, two different ways of define triangular neighbours are shown in Figure 1.6.

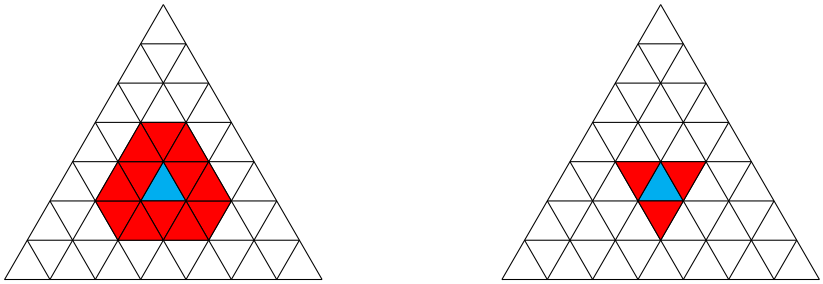


FIGURE 1.6. A cell (blue) and two possible ways of defining its neighbourhood (red) in a triangular tiling.

CHAPTER 2

Parity filter automata

Filter automata are a variation of cellular automata, first proposed in [8]. In a filter automaton, the value of a cell is dependent on both the values of cells from the previous time iteration, and the newly computed values of cells in the current time iteration. Specifically, for one-dimensional filter automata, cells are governed by the rule

$$a_i^{t+1} = \phi(a_{i-r}^{t+1}, \cdots a_{i-1}^{t+1}, a_i^t, \cdots, a_{i+r}^t), \tag{2.1}$$

with the restriction that $\phi(0, \cdots, 0) = 0$.

It is further assumed that the update rule is applied left to right and that there are only finitely many non-zero cells to the left.

As in Chapter 1.1, the parameter r is called the "radius" and specifies the size of the update window.

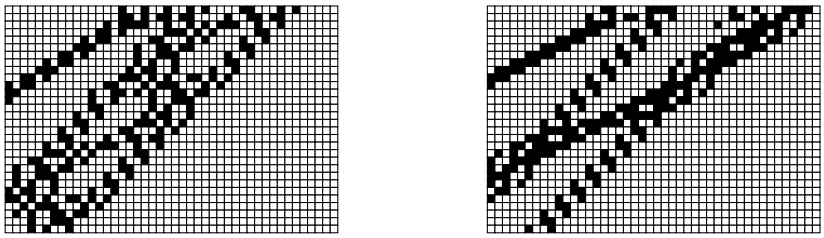
A one-dimensional filter automaton known as a "parity rule" filter automaton is studied in [8], in which cells are governed by the rule

$$a_i^{t+1} = \begin{cases} 1, & \text{if } S_i^{t+1} \geq 2 \text{ and is even,} \\ 0, & \text{otherwise,} \end{cases} \tag{2.2}$$

where

$$S_i^{t+1} := \sum_{n=i-r}^{i-1} a_n^{t+1} + \sum_{n=i}^{i+r} a_n^t. \tag{2.3}$$

Some examples of this rule applied to various initial configurations are shown in Figure 2.1. Following the diagrams in Chapter 1, white denotes a cell with value 0 and black a cell with value 1. Again, time flows downwards so that the top row is the initial configuration and subsequent rows represent successive iterations of the parity filter rule.



(A) (B)

FIGURE 2.1. Examples of the parity filter rule with $r = 3$.

In the remainder of this chapter, some terminology regarding the parity filter rule is established.

2.1. Particles

If there are enough 0 cells separating two regions of nonzero cells, then these two regions evolve independently of each other. We call these two independent regions "particles". See Figure 2.2 for an example.

The exact condition for when we may consider a configuration to contain multiple particles is given in Remark 3.6.

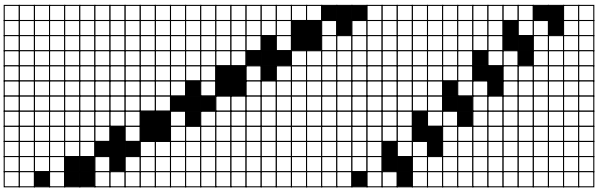


FIGURE 2.2. Two particles moving independently of each other ($r = 3$).

Hence, we often study the behaviour of a single particle, with the implication that the studied behaviour generalizes to configurations consisting of multiple particles, provided that they do not interfere with each other.

2.2. Periodicity

A particle may be periodic in the sense that the pattern of cells repeats after a fixed number of time iterations, possibly displaced by some number of units. The displacement d of a particle with period p can be measured by taking the difference between the index of the leftmost 1 at time $t + p$ and time t. We then define the speed of the particle to be

$$s := \frac{d}{p}. \tag{2.4}$$

Example 2.5. Consider the two particles in Figure 2.2. The left particle has period $p = 3$, displacement $d = 5$, and speed $s = \frac{5}{3}$. The right particle has period $p = 2$, displacement $d = 2$, and speed $s = 1$.

It is shown in [6] that the index of the leftmost 1 in any particle, excluding the particle $1\underbrace{0\cdots0}_{r}$ moves left by d units every time iteration for some d satisfying

$$0 \leq d \leq r - 1. \tag{2.6}$$

Hence, for a particle with given period p, we can upper bound its displacement over its period by $p(r-1)$.

2.3. Collisions

Two particles travelling at different speeds may collide in the sense that, from a certain point in time t_1, the two particles no longer evolve independently of each other.

A natural question is then to ask about the behaviour when two particles collide. Many collisions under the parity filter rule are solitonic in the sense that the original identity of the two particles are preserved following a collision. An example of such a collision is given in Figure 2.3.

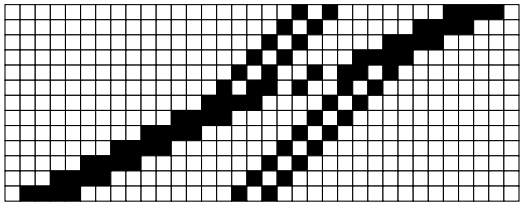


FIGURE 2.3. Solitonic collision of two particles ($r = 3$).

The behaviour here is similar to the behaviour of solitary wave solutions of nonlinear wave equations such as the Kortweg-de Vries equation, as noted in [9]. When two solitary waves travelling at different speeds collide with one another, the waves emerge from the collisions unchanged except for a phase shift. This is just like the collision of the two particles in Figure 2.3. In fact, a direct connection has been made between nonlinear wave equations and a different class of filter automata that also supports solitonic collisions [9].

2.4. Splitting

In some cases, a single particle may split into multiple independent particles if it evolves to a configuration where there is a long enough strip of contiguous 0 cells. An example of splitting is given in Figure 2.4.

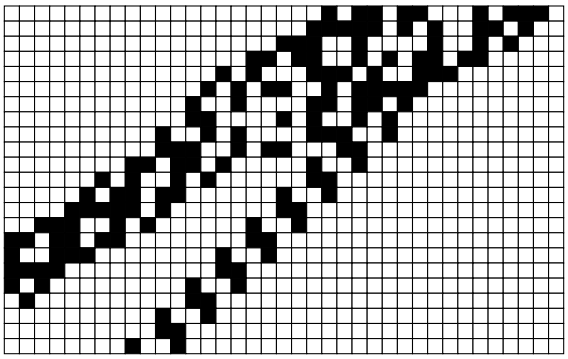


FIGURE 2.4. Splitting of a particle into two ($r = 3$).

A precise condition for when splitting occurs is given in Remark 3.6.

CHAPTER 3

Existing analytic results

Much work has been done in providing analytic results for the parity filter rule [6, 7]. These results are analytic in the sense that they explicitly describe the behaviour of the parity filter rule, as opposed to providing statistical properties observed through computer simulations.

A summary of the results in [6, 7] is presented in this chapter.

3.1. Fast rule theorem

An alternative characterization of the parity filter rule, known as the Fast Rule Theorem is presented in [6]. This result provides a much simpler way to compute successive iterations of the parity filter rule. To state the result, we first require the following definition.

Definition 3.1 (Box indices)**.** *Let the configuration a^t of the parity filter automaton at time t be given by*

$$a^t = \cdots 0\ a_0\ a_1\ a_2 \cdots$$

where $a_0 = 1$.

The box indices corresponding to a^t are defined using the following procedure:

(1) *The index of the first 1 is a box index (i.e. $i := 0$ is a box index).*
(2) *Indices following i in multiples of $r + 1$ are also box indices (i.e. $i, i + (r + 1), i + 2(r + 1), \cdots$ are box indices) up to and including the first box index that is followed by r consecutive 0s.*
(3) *If there are no more 1s to the right of the last box index, the process stops. Otherwise the process continues again from step 1 with i defined to be the next 1 to the right of the latest box index.*

Example 3.2. A configuration and its box indices, computed for $r = 3$ is given in Figure 3.1. It is assumed that there are only 0 cells on either side of the given configuration.

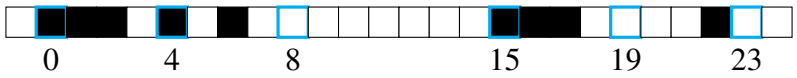


FIGURE 3.1. A configuration with its box indices highlighted in blue ($r = 3$).

The computation is performed as follows. Let the leftmost 1 (black cell) have index 0. Then by step 1, index 0 is a box index. Index $0 + (r + 1) = 4$ is also a box index by step 2, and so is index $4 + (r + 1) = 8$. Since there are $r + 1 = 4$ consecutive 0 cells following index 8, index 12 is *not* a box index by step 3. Instead, the process starts form step 1 with the next leftmost 1 which occurs at index 15. From here, indices 19 and 23 are box indices due to step 2. As there are no more 1s to the right of index 23, the process stops, and the set of box indices are $\{0, 4, 8, 15, 19, 23\}$.

The box indices play an important role in the Fast Rule Theorem which we now state below.

Theorem 3.3 (Fast Rule Theorem)**.**

Let B_t denote the box indices corresponding to the configuration at time t. Then the parity filter rule is equivalent to the rule

$$a_{j-r}^{t+1} = \begin{cases} a_j^t & \text{if } j \notin B_t \\ \overline{a_j^t} & \text{if } j \in B_t \end{cases} \tag{3.4}$$

where $\overline{x} := x + 1 \pmod 2$ is the complement of x.

This result can be proven by inducting over the cell indices. For a detailed proof, we refer the reader to [6].

Put simply, the configuration at time $t + 1$ can be computed by taking the configuration at time t, flipping the values of any cells falling on a box index, then shifting everything r units to the left.

Example 3.5. We return to the configuration in Example 3.2. By the Fast Rule Theorem, the next configuration is found by flipping the values of cells that fall in a box index, namely cells $0, 4, 8, 15, 19, 23$. Then, everything is shifted $r = 3$ units to the left. This is apparent in Figure 3.2.

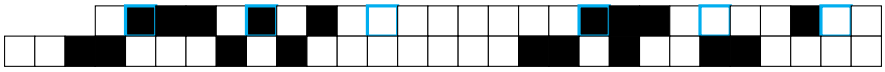


FIGURE 3.2. A configuration followed by one iteration of the parity filter rule ($r = 3$).

Remark 3.6. The final box index computed in step 2 which is followed by r consecutive 0s, marks the end of a particle. If there are any nonzero cells to the right of the r consecutive 0s, then these cells form part of a new particle that evolves separately from the particle to the left of the 0s.

This provides the precise condition for when two particles may be considered as independent particles, and when a single particles splits into multiple independent particles.

The Fast Rule Theorem provides the framework for many other analytic results which are presented below.

3.2. Stability

Often, it only makes sense to study cellular automata rules that are stable, in the sense that a finite number of nonzero cells evolve to a finite number of nonzero cells in subsequent time iterations. For instance, computer simulations are easy to perform if a rule is stable. Using the Fast Rule Theorem, one can easily show that the parity filter rule is indeed stable [6].

Theorem 3.7. *Parity filter automata are stable.*

Proof. If there are a finite number of 1s at time t then the set of box indices B_t is finite. Ignoring displacement, the configuration at time $t+1$ differs from the configuration at time t only over the finite set B_t hence it also consists of a finite number of 1s.

Since the initial configuration $t = 0$ contains a finite number of 1s, the statement holds by induction. □

3.3. 1-periodic particles

The Fast Rule Theorem can be used to construct a unique particle with period 1 and a given displacement d. Note that as a result of (2.6), we must have $0 \leq d \leq r-1$.

The construction relies on the following lemma, which was stated in [7], but with the full proof omitted. We state the lemma, give a detailed proof, and explain how it is used in the construction of 1-periodic particles.

Lemma 3.8. *Let $a^t = \cdots a_0^t\ a_1^t\ a_2^t \cdots$ be the configuration at time t and let B_t be the corresponding box indices. Then a^t is 1-periodic with displacement d if and only if*

$$a_{i+(r-d)}^t = \begin{cases} a_i^t, & \text{if } i+(r-d) \notin B_t, \\ \overline{a_i^t}, & \text{if } i+(r-d) \in B_t. \end{cases} \tag{3.9}$$

Proof. Suppose that a^t is 1-periodic with displacement d. By the Fast Rule Theorem applied on a_{i+r-d}^t in reverse, it follows that

$$a^t_{i+(r-d)} = \begin{cases} a^{t+1}_{i-d}, & \text{if } i+r-d \notin B_t, \\ \overline{a^{t+1}_{i-d}}, & \text{if } i+r-d \in B_t. \end{cases} \tag{3.10}$$

By periodicity, $a^{t+1}_{i-d} = a^t_i$ and so

$$a^t_{i+(r-d)} = \begin{cases} a^t_i, & \text{if } i+(r-d) \notin B, \\ \overline{a^t_i}, & \text{if } i+(r-d) \in B, \end{cases}$$

hence (3.9) holds.

Now suppose (3.9) holds. Again, apply the Fast Rule Theorem on $a^t_{i+(r-d)}$ in reverse to get (3.10). Equating (3.10) and (3.9), we conclude that $a^t_i = a^{t+1}_{i-d}$ i.e. that a^t is 1-periodic with displacement d. □

Using this lemma, we can construct a 1-periodic particle with displacement d as follows.

Start with $a_0 = 1$. Suppose $a_i = 1$ for some $i \in [1, \cdots, r-d-1]$. Then by the Fast Rule Theorem we have that $a_{i-r} = 1$ at the next time increment (since $0 < i < r+1$ is not a box index). This means that $i - r \leq -d - 1$ i.e. the leftmost 1 has moved at least $d+1$ cells to the left in the next time increment, which contradicts the particle a having displacement d. Hence we must have $a_i = 0$ for $i = 1, \cdots, r-d-1$ i.e. the first 1 is followed by $r-d-1$ number of 0s. Call the string $a_0 \cdots a_{r-d-1} = 10 \cdots 0$ the generating string. Note the case of $d = r-1$ results in the generating string of $a_0 = 1$.

Following (3.9), we must copy the generating string $r-d$ cells to the right, flipping the value of any cell that lie on a box index. We then copy this possibly modified generating string $r-d$ cells to the right again and flip any cell that lies on a box index, and so forth.

As it turns out, this construction eventually results in a box site being followed by r consecutive 0s. After this point, the pattern repeats, creating a new particle identical to the former particle, but which does not interact with the former particle. Hence we may stop the process here.

More formally, we have the following result. For a detailed proof, we refer the reader to [7].

Theorem 3.11 (Characterization of 1-periodic particles)**.** *Fix a displacement d where $0 \leq d \leq r-1$. Then there exists a unique 1-periodic particle*

$$a = a_0 \cdots a_L \tag{3.12}$$

which can be constructed as follows.

Let $M := lcm(r-d, r+1)$. Start with the string $a_0 \cdots a_{r-d-1} = 10 \cdots 0$ and copy this string $r-d$ cells to the right, flipping the value of a cell if it

lands on a box index. Take this (possibly modified) copy and again copy it $r-d$ *cells to the right, again flipping the value of a cell if it lands on a box index. Repeat this process until there is a box index with value* 0 *followed by* r *consecutive* 0*s.*

Specifically, after performing this procedure so that there are $\frac{2M}{r-d}$ *instances of the generating string or its modifications, the last box site will occur at index* $2M-(r+1)$*, have a value of* 0*, and be followed by* r *consecutive* 0*s, which signals the end of the process. The last* 1 *occurs at index* $L=2M-(r+1)-(r-d)$.

We now provide some examples to illustrate this construction.

Example 3.13. We construct the 1-periodic particle with displacement 1 for $r=3$. As $r-d-1=1$, the generating string is 10. Following the procedure described in (3.11) results in the particle shown in Figure 3.3.

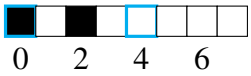


FIGURE 3.3. The unique 1-periodic particle with $d=1, r=3$. Box indices are highlighted in blue.

The construction is performed as follows. Assume the leftmost 1 is at index 0 as shown. The generating string 10 is copied into index 2 onwards, then again into index 4 onwards. However, since index 4 is a box index, the string is modified to 00. The modified string 00 is copied again into index 6 onwards. As the box index at index 4 is followed by $r=3$ consecutive 0s, the process stops.

Example 3.14. We construct the 1-periodic particle with displacement 2 for $r=4$. This particle is much longer than the previous example. As $r-d-1=2$, the generating string is 100. The construction described in (3.11) results in the particle given in Figure 3.4.

FIGURE 3.4. The unique 1-periodic particle with $d=2, r=4$. Box indices are highlighted in blue.

Corollary 3.15. *The unique fastest moving 1-periodic particle has displacement* $d=r-1$ *and is the particle* $a=\underbrace{1\cdots1}_{r+1}\underbrace{0\cdots0}_{r+1}$.

3.4. General evolution theorem

The Fast Rule Theorem describes the behaviour of the parity filter rule after a single time iteration. We now study the general evolutionary behaviour of the parity filter rule across multiple time iterations. To aid in the discussions in this section, we first require the following definition.

Definition 3.16. *A basic string is a string of* $r+1$ *bits* $a_0 \cdots a_r$ *where* $a_i \in \{0,1\}$ *for* $i = 0, \cdots, r$. *The set of basic strings is denoted by* X_{r+1} *and the zero basic string is* $0 := \underbrace{0 \cdots 0}_{r+1} \in X_{r+1}$.

Any configuration a^t can be written in terms of b-strings

$$a^t = \cdots 0 A_0^t \cdots A_\mu^t 0 \cdots \tag{3.17}$$

where $A_i \in X_{r+1}$ for $i = 0, \cdots, \mu$, the first bit of A_0^t is 1, and $A_\mu^t \neq 0$.

In the remainder of this section, we introduce a slight notational change as per [7] by shifting indices and cells r units to the right at each successive time iteration. In doing so, successive time iterations of the parity filter rule can be computed by simply copying the bits from the previous time iteration and flipping the values at all box indices. We also consider only single particles, and assume no splitting occurs.

The following example provides the motivation for the main result in this section.

Example 3.18. We first demonstrate the significance of the index shift and basic strings in Figure 3.5. An initial configuration and its three subsequent time iterations are shown.

The basic strings are delineated visually with spaces, so for example, the initial configuration as indicated by the top row, is the configuration

$$100010101010001,$$

where we have ignored the 0s to the left and right.

Box indices at each time step are highlighted in blue. The delineation of basic strings makes it easy to see that at any given time, the box indices fall within the same local index for each basic string.

Due to the shift in indexing, successive iterations of the parity filter rule can be computed by simply flipping the values of cells located at box indices.

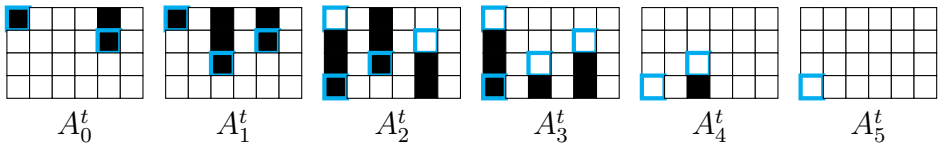


FIGURE 3.5. Iterations of the parity filter rule $r = 5$. Indices are shifted r units to the right each time iteration, basic strings are labelled and delineated with spaces, and box indices are highlighted in blue.

As time progresses, basic strings on the left evolve to the 0 string, and the original string reappears on the right. For instance the string A_0^0 evolves to $A_0^2 = 0$ by time $t = 2$, but at the same time $A_3^0 = 0$ evolves to $A_3^2 = A_0^0$.

To understand this more precisely, let $\ell(t)$ denote the local index of the first 1 in the leftmost nonzero string. Note that, as a result of the alignment of basic strings and the assumption that no splitting occurs, all box indices at time t occur at local index $\ell(t)$ within their respective strings.

For example, referring back to Figure 3.5, we have $\ell(0) = 0$ because the box indices at time $t = 0$ occur in local index 0. We also have $\ell(1) = 4$ because the box indices at time $t = 1$ occur in local index 4. Similarly, $\ell(2) = 2$ and $\ell(3) = 0$.

Let $f_i : X_{r+1} \longrightarrow X_{r+1}$ be the transformation that flips the bit in index i i.e. if $A = x_0 \cdots x_i \cdots x_r \in X_{r+1}$ then $f_i(A) = x_0 \cdots \overline{x_i} \cdots x_r \in X_{r+1}$.

Then the strings at time $t = 1$ are related to the strings at time $t = 0$ by

$$\begin{aligned} A_i^1 &= f_{\ell(0)}(A_i^0) \\ &= f_0(A_i^0) \end{aligned}$$

for $0 \leq i \leq 3$.

Similarly, for strings at time $t = 2$ we have

$$\begin{aligned} A_i^2 &= f_{\ell(1)} f_{\ell(0)}(A_i^0) \\ &= f_4 f_0(A_i^0) \end{aligned}$$

for $0 \leq i \leq 4$.

In other words, the strings A_i^2 for $0 \leq i \leq 4$ at time $t = 2$ can be computed as follows.

Consider the local indices corresponding to the cells with value 1 in the leftmost nonzero string at time $t = 0$, namely the string $A_0^0 = 100010$. These indices are $\{0, 4\}$.

To compute A_i^2, take the corresponding string A_i^0 at time $t = 0$ and flip the value of the cells falling on local indices $\{0, 4\}$.

With this process, it is evident that the string A_0^0 evolves to $A_0^2 = 0$ by time $t = 2$ since $f_4 f_0(100010) = 0$. Also, $A_3^0 = 0$ evolves to $A_3^2 = A_0^0$ by time $t = 2$ since $f_4 f_0(0) = 100010$.

Similar behaviour is evident between strings at time $t = 2$ and strings at time $t = 3$, namely that

$$A_i^3 = f_{\ell(2)}(A_i^2)$$

for $1 \leq i \leq 4$.

With analogous reasoning to before we see that A_1^2 evolves to $A_1^3 = 0$, and that A_4^2 evolves to $A_4^3 = A_1^2$. Furthermore, we see that A_3^2 evolves to $A_3^3 = A_1^0$.

In general, we see that whenever a string on the left is zeroed out (in this example at time $t = 2$ and $t = 3$), the corresponding string appears on the right.

This motivates the following definition.

Definition 3.19. *The landmark time τ_i is the smallest time for which the leftmost* 1 *is inside the basic string A_i.*

That is, at time $t = \tau_i$, we have that $A_j = 0$ for all $j < i$, and $A_i \neq 0$. Moreover this is the minimum such time that this occurs.

We define $\tau_0 := 0$.

Example 3.20. Following on from Example 3.18, the landmark times are $\tau_0 = 0, \tau_1 = 2, \tau_2 = 3$.

We can formalize the behaviour observed in Example 3.18 as follows.

Let the configuration of the automaton at time t be

$$a^t = \cdots 0\ A_k^t \cdots A_\mu^t\ 0 \cdots$$

Let $\ell(t)$ denote the local index of the first 1 in A_k^t and let $\lambda(t)$ denote the local index of the last 1 in A_μ^t. Let $f_i : X_{r+1} \longrightarrow X_{r+1}$ be defined as above.

Then the following holds, provided that no splitting occurs:

$$a^{t+1} = \begin{cases} \cdots 0\ f_{\ell(t)}(A_0^t) \cdots f_{\ell(t)}(A_\mu^t)\ f_{\ell(t)}(0)\ 0 \cdots & \text{if } \ell(t) < \lambda(t) \\ \cdots 0\ f_{\ell(t)}(A_0^t) \cdots f_{\ell(t)}(A_\mu^t)\ 0 \cdots & \text{if } \ell(t) \geq \lambda(t) \end{cases} \tag{3.21}$$

Furthermore, it turns out the condition $\ell(t) < \lambda(t)$ occurs if and only if t is a landmark time.

Through repeated application of (3.21), one can show the following result, which predicts the general evolutionary behaviour of a particle. For a detailed proof, we refer the reader to [7].

Theorem 3.22 (General Evolution Theorem)**.** *Let the initial configuration of a single particle*

$$a^0 = \cdots 0 A_0^0 \cdots A_\mu^0 0 \cdots$$

be defined so that A_0^0 *begins with a 1,* $A_i^0 \neq 10\cdots 0$ *for* $i = 0, \cdots, \mu$ *and* $A_\mu^0 \neq 0$. *Define*

$$F_0 := I, \quad F_t := f_{\ell(t-1)} \cdots f_{\ell(0)} \text{ for } t > 0$$

where $I : X_{r+1} \longrightarrow X_{r+1}$ *is the identity transformation.*

Then the configuration at time $t = T$ *where* $\tau_n < T \leq \tau_{n+1}$ *is*

(3.23)
$$a^T = \cdots 0\ F_T(A_n^0)\ F_T(A_{n+1}^0) \cdots F_T(A_\mu^0) \quad F_{\tau_0}F_T(0)\ F_{\tau_1}F_T(0) \cdots F_{\tau_{n-1}}F_T(0)\ 0 \cdots .$$

In particular, the configuration at landmark times $t = \tau_n$ *is*

(3.24)
$$a^{\tau_n} = \cdots 0\ F_{\tau_n}(A_n^0)\ F_{\tau_n}(A_{n+1}^0) \cdots F_{\tau_n}(A_\mu^0) \quad A_{n-1}^{\tau_0}\ A_{n-1}^{\tau_1} \cdots A_{n-1}^{\tau_{n-1}}\ 0 \cdots .$$

It is assumed throughout that no splitting occurs for $0 < t < T$ *and that the initial configuration* $t = 0$ *is advanced if necessary to ensure that* $0 \leq n \leq \mu$.

An easy consequence of the general evolution theorem is the following result.

Theorem 3.25. *Any particle consisting of one basic string* $A_0^0 \neq 10\cdots 0$ *is periodic with period* $p \leq \tau_1$, *the number of 1s in* A_0^0.

Proof. Let $a^0 = \cdots 0\ A_0^0\ 0 \cdots$. Since there are at least two 1s in A_0^0, no splitting occurs, hence we can apply the General Evolution Theorem at time $t = \tau_1$ to obtain

$$\begin{aligned} a^{\tau_1} &= \cdots 0\ F_{\tau_1}(A_0^0)\ F(0)\ 0 \cdots \\ &= \cdots 0\ 0\ A_0^0\ 0 \cdots . \end{aligned} \tag{3.26}$$

□

CHAPTER 4

Basic strings and a generalized parity filter rule

In this chapter we study a generalization of the parity filter rule. To the best of our knowledge, the results shown here are original.

We initially studied the rule given by

$$a_i^{t+1} = \begin{cases} 1, & \text{if } S_i^{t+1} \geq 4 \text{ and is even,} \\ 0, & \text{otherwise.} \end{cases} \tag{4.1}$$

This was motivated by brief comments made in [8] that particles under this rule demonstrate similar behaviour to the original parity filter rule.

We derived a series of results under this rule that turned out to also hold for the more general rule given by

$$a_i^{t+1} = \begin{cases} 1, & \text{if } S_i^{t+1} \geq 2q \text{ and is even,} \\ 0, & \text{otherwise.} \end{cases} \tag{4.2}$$

We refer to this as the generalized parity filter rule, which now has two parameters q, r. The results we derived for the generalized parity filter rule are presented in this section.

Note that the rule in (4.1) is a special case where $2q = 4$. Furthermore, the original parity filter rule is a special case where $2q = 2$.

4.1. Alternative characterization

Our main result is an alternative characterization of the generalized parity filter rule (4.2) for configurations described by basic strings. Recall that a basic string is a sequence of bits of length $r + 1$. We further assume that the first element of the string is nonzero.

We show that the evolution of basic strings under the generalized parity filter rule can be computed as a limited cyclical shift, and that all basic strings are periodic. We provide an explicit formula to calculate the period and displacement of any basic string.

Before stating the result, we first provide a few examples of basic strings and their time evolution, as given in Figure 4.1.

(A) (B)

FIGURE 4.1. Evolution of two different basic strings under the rule $2q = 4, r = 6$.

Notice that from a given configuration at time t, we can compute the configuration at the next time iteration $t + 1$ as follows:

(1) Copy the configuration of cells at time t into that of time $t + 1$.
(2) Take the first three cells with value 1 and place them at the end of the string i.e. so that a_0 is at index $r + 1 = 7$, a_1 is at index $r + 2 = 8$, etc. assuming that a_0 is the first nonzero cell.
(3) Move everything $r = 6$ units to the left.

In general, the following result holds.

Theorem 4.3. *Suppose the configuration at time t is given by*

$$a_t = \cdots 0\ a_0 \cdots a_r\ 0 \cdots$$

where $a_0 = 1$ and $\sum_{i=0}^{r} a_i \geq 2q$. Let j denote the index of the $(2q-1)$st 1 i.e. $a_j = 1$ and $\sum_{i=0}^{j} a_i = 2q - 1$.

Then the configuration at time $t + 1$ produced by the generalized parity filter rule is given by

$$a_i^{t+1} = \begin{cases} a_{i+r}^t, & \text{if } j - r + 1 \leq i \leq 0, \\ a_{i-1}^t, & \text{if } 1 \leq i \leq j + 1, \\ 0, & \text{otherwise i.e. if } i \leq j - r \text{ or } i \geq j + 2. \end{cases} \tag{4.4}$$

In other words, the configuration at time $t + 1$ can be found as follows:

(1) *Copy the configuration of cells at time t into that of time $t + 1$.*
(2) *Take the first $2q - 1$ cells with value 1 and place them at the end of the string i.e. so that a_0 is at index $r + 1$, a_1 is at index $r + 2$, etc.*
(3) *Move everything r units to the left.*

Proof. First, note that since there are at least $2q$ number of 1s, we must have $j \leq r - 1$.

We will make repeated reference to the update window $W(i)$ defined as

$$\begin{array}{c} a_i^t \cdots a_{i+r}^t \\ a_{i-r}^{t+1} \cdots a_i^{t+1}. \end{array}$$

We split the proof case by case.

Case 1: $a_i^{t+1} = 0$ for $i \leq j - r$.

For $i \ll j - r$ we have $S_i^{t+1} = 0$ hence (4.4) holds. Now assume (4.4) holds for all $i' \leq i$ where $i \leq j - r - 1$. Consider the sum S_{i+1}^{t+1} corresponding to the cell a_{i+1}^{t+1}, given by

$$\begin{aligned} S_{i+1}^{t+1} &= \sum_{n=i+1-r}^{i} a_n^{t+1} + \sum_{n=i+1}^{i+r+1} a_n^t \\ \text{(by inductive hypothesis)} \qquad &= 0 + \sum_{n=i+1}^{i+r+1} a_n^t \\ \text{(since } i + r + 1 \leq j\text{)} \qquad &\leq 2q - 1. \end{aligned}$$

Hence $a_{i+1}^{t+1} = 0$ and (4.4) holds.

Case 2: $a_i^{t+1} = a_{i+r}^t$ for $j - r + 1 \leq i \leq 0$.

We will show that for all $j - r + 1 \leq i \leq 0$ the number of 1s in the update window $W(i)$ (given by $S_i^{t+1} + a_i^{t+1}$) is odd and greater than or equal to $2q - 1$.

First, consider if $i = j - r + 1$. The sum S_{j-r+1}^{t+1} corresponding to cell a_{j-r+1}^{t+1} is given by

$$\begin{aligned} S_{j-r+1}^{t+1} &= \sum_{n=j-2r+1}^{j-r} a_n^{t+1} + \sum_{n=j-r+1}^{j+1} a_n^t \\ \text{(since } a_n^t < 0 \text{ for } n < 0\text{)} \qquad &= 0 + \sum_{n=j-r+1}^{j+1} a_n^t \\ \text{(by case 1)} \qquad &= 0 + \sum_{n=0}^{j+1} a_n^t \\ \text{(by definition of } j\text{)} \qquad &= 2q - 1 + a_{j+1}^t. \end{aligned}$$

If $a_{j+1}^t = 1$ then $S_{j-r+1}^{t+1} = 2q$ hence $a_{j-r+1}^{t+1} = 1 = a_{j+1}^t$ and (4.4) holds. The number of 1s in the update window $W(j - r + 1)$ is $2q + 1 \geq 2q - 1$ and $2q + 1$ is odd. On the other hand, if $a_{j+1}^t = 0$ then $S_{j-r+1}^{t+1} = 2q - 1$ and so $a_{j-r+1}^{t+1} = 0 = a_{j+1}^t$ and (4.4) holds. The number of 1s in the update window $W(j - r + 1)$ is $2q - 1 \geq 2q - 1$ and $2q - 1$ is odd.

Now suppose that for some i where $j - r + 1 \leq i \leq -1$, the number of 1s in the update window $W(i)$ is odd and greater or equal to $2q - 1$, and that (4.4) holds. Consider the combined update windows $W(i)$ and $W(i + 1)$ as follows:

$$\begin{array}{cc} a_i^t & \boldsymbol{a_{i+1}^t \cdots a_{i+r}^t}\ a_{i+r+1}^t \\ a_{i-r}^{t+1}\ \boldsymbol{a_{i-r+1}^{t+1} \cdots a_i^{t+1}} & a_{i+1}^{t+1}. \end{array}$$

The region shared by both $W(i)$ and $W(i+1)$ is bolded. The update windows differ only in that for $W(i+1)$, the cells a_i^t and a_{i-r}^{t+1} are replaced with a_{i+r+1}^t and a_{i+1}^{t+1}. The 1s in the update window $W(i)$ cannot appear in the cells a_i^t and a_{i-r}^{t+1}. To see this, first observe that $i \leq -1$ so $a_i^t = 0$. Then also $a_{n-r}^{t+1} = 0$ whether by case 1 or case 2. Hence the 1s in $W(i)$ must lie in the shared (bolded) region. It follows that $S_{i+1}^{t+1} - a_{i+r+1}^t \geq 2q-1$ and is odd.

Similar to before, if $a_{i+r+1}^t = 1$ then $S_{i+1}^{t+1} \geq 2q$ is even and so $a_{i+1}^{t+1} = 1 = a_{i+r+1}^t$. The number of 1s in $W(i+1)$ is odd and greater or equal to $2q+1 \geq 2q-1$. On the other hand, if $a_{i+r+1}^t = 0$ then $S_{i+1}^{t+1} \geq 2q-1$ is odd and so $a_{i+1}^{t+1} = 0 = a_{i+r+1}^t$. The number of 1s in $W(i+1)$ is odd and greater or equal to $2q-1$.

Case 3: $a_i^{t+1} = a_{i-1}^t$ for $1 \leq i \leq j+1$.

Here the sum S_i^{t+1} corresponding to cell a_i^{t+1} is given by

$$\begin{aligned} S_i^{t+1} &= \sum_{n=i-r}^{i-1} a_n^{t+1} + \sum_{n=i}^{i+r} a_n^t \\ \text{(by case 1)} \qquad &= \sum_{n=j-r+1}^{i-1} a_n^{t+1} + \sum_{n=i}^{i+r} a_n^t \\ \text{(by case 2)} \qquad &= \sum_{n=j+1}^{r} a_n^t + \sum_{n=1}^{i-1} a_n^{t+1} + \sum_{n=i}^{i+r} a_n^t \\ \text{(since } a_n^t = 0 \text{ for } n \geq r+1) \qquad &= \sum_{n=j+1}^{r} a_n^t + \sum_{n=1}^{i-1} a_n^{t+1} + \sum_{n=i}^{r} a_n^t. \end{aligned}$$

If $i = 1$ then we have

$$\begin{aligned} S_1^{t+1} &= \sum_{n=j+1}^{r} a_n^t + \sum_{n=1}^{r} a_n^t \\ &= \sum_{n=1}^{j} a_n^t + 2\sum_{n=j+1}^{r} a_n^t \\ &= (2q-2) + 2\sum_{n=j+1}^{r} a_n^t. \end{aligned}$$

Since $\sum_{n=0}^{r} a_n^t = 2q$ we have that $\sum_{n=j+1}^{r} \geq 1$ by definition of j and so $S \geq 2q$ and is even. Hence $a_1^{t+1} = 1 = a_0^t$ and so (4.4) holds for $i = 1$.

Now assume (4.4) holds for all $i' \leq i$ for some i where $1 \leq i \leq j$. Then we have

$$\begin{aligned} S_{i+1}^{t+1} &= \sum_{n=j+1}^{r} a_n^t + \sum_{n=1}^{i} a_n^{t+1} + \sum_{n=i+1}^{r} a_n^t \\ \text{(by case 3)} \qquad &= \sum_{n=j+1}^{r} a_n^t + \sum_{n=0}^{i-1} a_n^t + \sum_{n=i+1}^{r} a_n^t \\ &= \left(\sum_{n=0}^{i-1} a_n^t + \sum_{n=i+1}^{j} a_n^t \right) + 2 \sum_{n=j+1}^{r} a_n^t \\ &= (2q - 1 - a_i^t) + 2 \sum_{n=j+1}^{r} a_n^t. \end{aligned}$$

If $a_i^t = 0$ then S_{i+1}^{t+1} is odd so $a_{i+1}^{t+1} = 0 = a_i^t$ and (4.4) holds. On the other hand, if $a_i^t = 1$ then S_{i+1}^{t+1} is even, and moreover, since $\sum_{n=j+1}^{r} a_n^t \geq 1$ we must have that $S_{i+1}^{t+1} \geq 2q$. Hence $a_{i+1}^{t+1} = 1 = a_i^t$ and (4.4) holds.

Case 4: $a_i^{t+1} = 0$ for $j + 2 \leq i \leq r$.

Here we have

$$\begin{aligned} S_i^{t+1} &= \sum_{n=i-r}^{i-1} a_n^{t+1} + \sum_{n=i}^{i+r} a_n^t \\ \text{(since } a_n^t = 0 \text{ for } n \geq r+1\text{)} \qquad &= \sum_{n=i-r}^{i-1} a_n^{t+1} + \sum_{n=i}^{r} a_n^t \\ \text{(by case 2)} \qquad &= \sum_{n=i}^{r} a_n^t + \sum_{n=1}^{i-1} a_n^{t+1} + \sum_{n=i}^{r} a_n^t \\ \text{(by case 3)} \qquad &= \sum_{n=i}^{r} a_n^t + \sum_{n=0}^{j} a_n^t + \sum_{n=j+2}^{i-1} a_n^{t+1} + \sum_{n=i}^{r} a_n^t \\ &= 2 \sum_{n=i}^{r} a_n^t + (2q - 1) + \sum_{n=j+2}^{i-1} a_n^{t+1}. \end{aligned}$$

If $i = j + 2$ then $S_i^{t+1} = 2q - 1 + 2 \sum_{n=i}^{r} a_n^t$ which is odd, hence $a_i^{t+1} = 0$ and (4.4) holds.

Now assume (4.4) holds for all $i' \leq i$ for some i where $j+2 \leq i \leq r-1$. Then we have

$$\begin{aligned} S_{i+1}^{t+1} &= 2 \sum_{n=i+1}^{r} a_n^t + (2q-1) + \sum_{n=j+2}^{i} a_n^{t+1} \\ &= 2 \sum_{n=i+1}^{r} a_n^t + (2q-1) + 0 \end{aligned}$$

which is again odd, hence $a_i^{t+1} = 0$ and (4.4) holds.

Case 5: $a_i^{t+1} = 0$ for $r+1 \leq i$.

Here we have

$$\begin{aligned} S_i^{t+1} &= \sum_{n=i-r}^{i-1} a_n^{t+1} + \sum_{n=i}^{i+r} a_n^t \\ (a_n^t = 0 \text{ for } n \geq r+1) \qquad &= \sum_{n=i-r}^{i-1} a_n^{t+1} \\ \text{(by case 3)} \qquad &= \sum_{n=i-r-1}^{j} a_n^t + \sum_{n=j+2}^{i-1} a_n^{t+1} \\ \text{(by case 4)} \qquad &= \sum_{n=i-r-1}^{j} a_n^t + 0 + \sum_{n=r+1}^{i-1} a_n^{t+1} \end{aligned}$$

If $i = r+1$ then $S_i^{t+1} = \sum_{n=0}^{j} a_n^{t+1} = 2q-1$ hence $a_i^{t+1} = 0$ and (4.4) holds.

Now assume (4.4) holds for all $i' \leq i$ for some i where $r+1 \leq i$. Then

$$\begin{aligned} S_i^{t+1} &= \sum_{n=i-r}^{j} a_n^{t+1} + \sum_{n=r+1}^{i} a_n^{t+1} \\ &\leq (2q-1) + 0, \end{aligned}$$

hence $a_i^{t+1} = 0$ and (4.4) holds. □

Remark 4.5. The procedure described in (4.4) can be thought of as a limited cyclical shift by r units to the left, in the following sense.

(1) All cells are shifted to the left one unit at a time up to a total of r units, subject to the following constraint.
(2) The first $(2q-1)$ 1s that are shifted to the left of index 0 are removed and reappear on the right at index r. Subsequent 1s shift to the left of index 0 as normal.

4.2. Periodicity

Using (4.4) we will show that all basic strings are periodic under the generalized parity filter rule.

Firstly, it is easy to see that the procedure described in (4.5) does not create or remove any 1s, hence the following result holds.

Corollary 4.6. *The number of 1s in the evolution of a basic string is constant.*

Note that (4.6) already implies that all basic strings are periodic as there are a finite number of 1s and only a finite number of ways to rearrange them. However, we can provide a stronger bound on the period, as we state in the following result.

Corollary 4.7. *A basic string is periodic with period $p \leq k$.*

Proof. Since altering the relative positions of two strings does not change their periodicity, we may ignore the first step in (4.5) and only consider the second step. After k iterations of the second step, a total of $(2q-1)k$ number of 1s have been shifted to the left of index 0 and have reappeared on the right. Since there are k number of 1s in the string, the resulting string must be the same as the original string. □

Remark 4.8. In the proof of Corollary 4.7, we only require that a multiple of k number of 1s be shifted past the left of index 0 in order for the original string to reappear. Hence at any $t > 0$ satisfying

$$(2q-1)t \equiv 0 \mod k, \tag{4.9}$$

the original string will reappear. The smallest $t > 0$ satisfying (4.9) is given by

$$t = \frac{k}{\gcd(k, 2q-1)}, \tag{4.10}$$

hence we may take the period to be $p \leq \frac{k}{\gcd(k,2q-1)}$.

Note that, however, $p = \frac{k}{\gcd(k,2q-1)}$ is not necessarily the minimal period. It may be the case that at some $t' < \frac{k}{\gcd(k,2q-1)}$, we have $(2q-1)t' \not\equiv 0 \mod k$ yet the original string reappears, due to symmetry in the string.

An example of this with the rule $2q = 4, r = 4$ is illustrated in Figure 4.2. The particle has period $1 < \frac{k}{\gcd(k,2q-1)} = 5$.

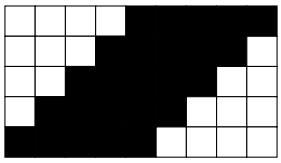


FIGURE 4.2. An example of the evolution of a basic string under the rule $2q = 4, r = 4$ where the period $p < \frac{k}{\gcd(k,2q-1)}$.

4.3. Displacement

We now study the displacement of a basic string over its period. Let $a = a_0 \cdots a_r$ be a basic string where $a_0 = 1$ with period $p = k := \sum_{i=0}^{r} a_i \geq 2q$. Let j_n for $0 \leq n \leq k-1$ denote the index of the $(n+1)$st 1 i.e. $a_{j_n} = 1$ and $\sum_{i=0}^{j_n} a_i = n+1$. For ease of notation later on, define $j_n := j_{n \mod k}$ for $n \geq k$. Define

$$f(x,y) := \begin{cases} j_y - j_x, & \text{if } j_y \geq j_x, \\ j_y - j_x + (r+1), & \text{otherwise.} \end{cases}$$

In other words, $f(x,y)$ counts the number of cells required to start from the $(x+1)$st 1, and move right to reach the $(y+1)$st 1 such that the cell indices are treated as mod $r+1$ i.e. counting past the right of index r wraps back around to index 0.

Let d_{t+1} denote the left displacement of the leftmost 1 from time t to time $t+1$. During this time, all cells have been shifted r units to the left, except the first $(2q-1)$ 1s which have been cycled to the back. Hence the ith 1 at time t corresponds to the $(i+(2q-1)t)$th 1 in the original string, assuming that counting past the kth 1 loops back to the start of the string. The leftmost 1 at time $t+1$ corresponds to the fourth 1 at time t which is $f((2q-1)t, (2q-1)(t+1))$ cells to the right of the leftmost 1 at time t. Hence $d_{t+1} = r - f((2q-1)t, (2q-1)(t+1))$.

Since a is periodic with period k then the configuration at time $t = 0$ and $t = k$ must be the same, ignoring any displacement. The displacement d can then be calculated by the displacement of the leftmost 1 between these

two points in time which is given by

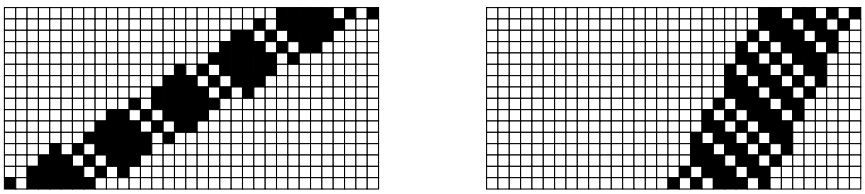
$$\begin{aligned} d &= \sum_{t=1}^{k} d_t \\ &= \sum_{t=1}^{k} (r - f((2q-1)(t-1), (2q-1)t)) \\ &= rk - \sum_{t=1}^{k} f((2q-1)(t-1), (2q-1)t) \\ &= rk - (2q-1)(r+1). \end{aligned}$$

The equality $\sum_{t=1}^{k} f((2q-1)(t-1), (2q-1)t) = (2q-1)(r+1)$ follows from the fact that there are k number of 1s distributed across a string of length $r+1$.

This proves the following.

Theorem 4.11. *Let $a = a_0 \cdots a_r$ be a basic string with $a_0 = 1$ and $k := \sum_{i=0}^{r} a_i \geq 2q$. Under the generalized parity filter rule, the particle a is periodic with period $p = k$ and displacement $d = rk - (2q-1)(r+1)$.*

Example 4.12. Two examples of basic strings under the rule $2q = 6, r = 8$ are given in Figure 4.3.



(A) (B)

FIGURE 4.3. Two examples of basic strings under the rule $2q = 6, r = 8$.

In Figure 4.3a we have $k = 7$ hence by (4.11) the particle is periodic with period $p = 7$ and left displacement $8 \cdot 7 - 5 \cdot 9 = 11$. In Figure 4.3b we have $k = 6$ hence the particle is periodic with period $p = 6$ and left displacement $8 \cdot 6 - 5 \cdot 9 = 3$. The reader is invited to verify these results.

4.4. Connection to the original parity filter rule

The special case of $2q = 2$ gives the original parity filter rule. Applying (4.11) for $2q = 2$ gives an alternate proof of (3.25). We can further calculate

the displacement of the string as $d = rk - (r + 1)$, which agrees with the displacement observed in (3.26).

We also see that (4.3) for $2q = 2$ is a special case of the Fast Rule Theorem applied to basic strings.

Bibliography

[1] C. Bays. "Cellular Automata in Triangular, Pentagonal and Hexagonal Tessellations". *Encyclopedia of Complexity and Systems Science*. Ed. by R. A. Meyers. New York, NY: Springer New York, 2009, pp. 892–900. ISBN: 978-0-387-30440-3. DOI: 10.1007/978-0-387-30440-3_58. URL: https://doi.org/10.1007/978-0-387-30440-3_58.

[2] J. Conway. "The game of life". *Scientific American* 223.4 (1970), p. 4.

[3] M. Cook. "Universality in elementary cellular automata". *Complex systems* 15.1 (2004), pp. 1–40.

[4] J. Greenberg, B. Hassard, and S. Hastings. "Pattern formation and periodic structures in systems modeled by reaction-diffusion equations". *Bulletin of the American Mathematical Society* 84.6 (1978), pp. 1296–1327.

[5] C. Ormerod. "Cellular automata model of HIV infection on tilings of the plane". *Proceedings of the 7th Asia-Pacific Conference on Complex Systems*. 2004.

[6] T. Papatheodorou, M. Ablowitz, and Y. G. Saridakis. "A rule for fast computation and analysis of soliton automata". *Studies in Applied Mathematics* 79.2 (1988), pp. 173–184.

[7] T. Papatheodorou and A. Fokas. "Evolution theory, periodic particles, and solitons in cellular automata". *Studies in Applied Mathematics* 80.2 (1989), pp. 165–182.

[8] J. K. Park, K. Steiglitz, and W. P. Thurston. "Soliton-like behavior in automata". *Physica D: Nonlinear Phenomena* 19.3 (1986), pp. 423–432.

[9] T. Tokihiro et al. "From soliton equations to integrable cellular automata through a limiting procedure". *Physical Review Letters* 76.18 (1996), p. 3247.

[10] S. Wolfram. *A new kind of science*. Vol. 5. Wolfram media Champaign, IL, 2002.

[11] S. Wolfram. "Cryptography with Cellular Automata". *Advances in Cryptology — CRYPTO '85 Proceedings*. Ed. by H. C. Williams.

Springer Berlin Heidelberg, 1986, pp. 429–432. ISBN: 978-3-540-39799-1.

[12] S. Wolfram. “Random sequence generation by cellular automata”. *Advances in applied mathematics* 7.2 (1986), pp. 123–169.

[13] S. Wolfram. “Statistical mechanics of cellular automata”. *Reviews of modern physics* 55.3 (1983), p. 601.

[14] J. T. Wootton. “Local interactions predict large-scale pattern in empirically derived cellular automata”. *Nature* 413.6858 (2001), pp. 841–844.