Published in: Journal of Technical Physics, v.36, No.2 (1995), pp. 153-163.

Filter automata and PRR filtering

Paweł Siwak
Poznan University of Technology, 60-965 Poznan, Poland
(*e-mail*: psiw@agat.sk-kari.poz.edu.pl)

Abstract

Parity rule recursive (PRR) filtering was defined by Steiglitz and others and shown to support certain moving periodic strings (called particles) with soliton-like properties. Particles are easily recognized on space-time diagrams. Later, some meaningful computations were shown to be performed by colliding particles. The aim of this paper is to present the family of filter automata (FA) and to show that they constitute a generalization of PRR filtering. We define FA by applying the notions of an activating function, a program, and a reset function. These facilitate description of transmitting media by a pipeline of FA, that is by a one-way linear cellular automaton. Our family of FA supports not only particles of PRR filtering but also other types of periodic and non-periodic strings. For example, contrary to PRR filtering, the FA support annihilation and strange particles. We give some examples of FA behavior.

Key words: filter automata, cellular automata, iterative computations, polyautomata, pipelined arrays, discrete waves, solitons, particles.

1. Introduction

In 1986, Park, Steiglitz and Thurston [15] introduced the so called parity rule filter automata (PRFA). PRFA are transducers, that is they constitute the sequential model of processing the discrete sequences. In signal processing such transducers are called parity rule recursive filters (briefly PRR filtering) and are classified as infinite impulse response (IIR) filters [4]. Throughout this paper we will use the name PRR filtering.

The PRR filtering, when applied iteratively to a sequence a^t, exhibits on parallelograms (space-time diagrams) certain moving periodic strings, which are called particles. Particles are of great importance in computations and in simulation of processes. They demonstrate soliton-like behavior during collisions. Moreover, they seem to be necessary in global computations where their role is to carry and spread local results. It allows coordination of final results [2, 3, 8, 9]. A number of papers have been devoted to the analysis of particle behavior [1, 5, 6, 7, 13]. It was even shown that some meaningful computations can be performed by phase coding particles [20]. Although the behavior of particles of PRR filtering was extensively studied, there emerges the general and rather pessimistic conclusion that some new notions are necessary to allow for an analytical treatment of evolution of particles [12, 13]. It is nearly impossible to follow analytically the operation of PRR filtering and predict its behavior, if one only uses the parity rule itself. As a remedy, the Fast Rule Theorem, an equivalent explicit formulation of PRR filtering, has been proposed [12]. Nevertheless, fundamental impediments in particle analysis still remain. The introduction of the family of filter automata (briefly FA) is our solution to this problem. In the earlier introductory paper [18] it was shown that this new description of sequential transducers, which uses operations instead of states, has the potential to advance the study of particles. Here we present a formal definition of FA and show how they generalize the PRR filtering. We also define precisely the particle itself. In [13] it was pointed out that deciding under what conditions two strings must be considered as a multiple or a single particle, is either left open or not answered precisely. To resolve this problem we introduce here the notion of an internal idle step of FA which allows to reveal all connected substrings of any discrete sequence. This means that we propose to use our FA model as a classifier of sequences.

A comment is also necessary here on the relation of the models of PRR filtering and of FA to cellular automata (CA). It must be emphasized that PRR filtering as well as the FA are sequential models, whereas cellular automata operate by definition in parallel, that is all elements of discrete sequence are updated in CA simultaneously. Nevertheless, when particles are under consideration we use the sequential models as iterating devices. As such they were studied in [21] and shown to be computationally more powerful than finite machines used as decision devices. A single iterating device, e.g., a Mealy machine, may be replaced by a pipeline of its copies, if some simple conditions are fulfilled. This leads us to the model known as a one-way cellular automaton, that is a linear CA with only one (left-hand) neighbor. Another name of it is OLIA (one-way linear iterative automaton) [8]. This model has been extensively studied [2, 9] and it is clear now that it is computationally at least as powerful as a universal Turing machine [17]. This means that a linear pipe of some Mealy machines used as a processing node is capable of performing any computation (in the sense of TM).

From the engineering point of view, the relevant problem is to decide which subclass of Mealy machines should be chosen to play the role of a universal processing node in linear structures. Systolic processing with the nodes executing inner product is one possible answer [16]. Our answer given here is the family of FA. This is a simple, well-defined subclass of finite state machines. The power and elegance of this family lie in the fact that we define it using not the notion of a state of processing machine (which is typical in automata theory), but instead, making use of the notions of activation, operations and extinction of processing. These notions correspond to the interaction between a transmitting medium and data when they are passed through. We believe that they will help understand better the phenomenon of soliton. The activation, operations and extinction of processing are not only natural but also easy to control, which is especially important if one thinks of using the transmitting media to act as a linear pipe of processors. In this way the FA model brings us closer to the idea of using optical fibers in the double role: to transmitting as well as to processing of information.

2. Definition of a filter automaton

A filter automaton (briefly FA) is a formal model capable of processing sequences over a finite alphabet Σ. We denote it by $F = (\Sigma, \iota, \rho, \kappa)$ and define it as follows:

$\Sigma = \{0, 1, ..., m\}$ is a finite set, called the alphabet of F, with $m > 0$,

$\iota : \Sigma \rightarrow \{0, 1\}$ is a mapping, called the activating function of F,

$\rho : \Sigma^k \rightarrow \{0, 1\}$ is a mapping, called the reset function of F, with $k > 1$,

$\kappa = (f_0, f_1, \dots, f_k)$ is a sequence of mappings $f_i : \Sigma \rightarrow \Sigma$, called the program of F, which generates outputs in discrete time steps τ according to the following order:

– $\mathrm{next}(f_0(\sigma_\tau)) = \begin{cases} f_0(\sigma_{\tau+1}) & \text{if } \iota(\sigma_\tau) = 0 \\ f_1(\sigma_{\tau+1}) & \text{if } \iota(\sigma_\tau) = 1 \end{cases}$

– $\mathrm{next}(f_i(\sigma_\tau)) = f_{i+1}(\sigma_{\tau+1})$ for $0 < i < k$

$$- \quad \text{next}(f_k(\sigma_\tau)) = \begin{cases} f_1(\sigma_{\tau+1}) & \text{if } \rho(\sigma_{\tau-k+1}, \sigma_{\tau-k+2}, \ldots, \sigma_\tau) = 0 \\ f_0(\sigma_{\tau+1}) & \text{if } \rho(\sigma_{\tau-k+1}, \sigma_{\tau-k+2}, \ldots, \sigma_\tau) = 1 . \end{cases}$$

The operation of a filter automaton is illustrated in Figure 1.

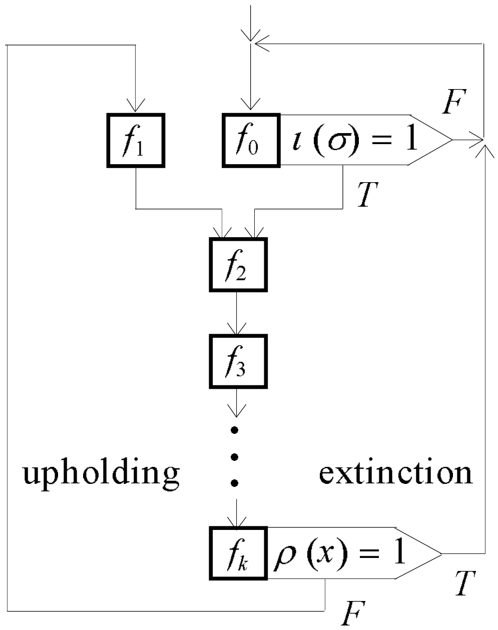

Fig. 1. Flow chart of the filter automaton (activation, operations and extinction).

A filter automaton operates in two modes, it is either inactive or active. An inactive filter automaton waits for the occurrence of an activating element (such that $\iota(\sigma) = 1$) while the read symbols are output as $f_0(\sigma)$. We assume $0 \in \Sigma$ to be the quiescent symbol, which means that $f_0(0) = 0$.

When the filter automaton is activated, successive operations of its program are executed; we say that the automaton performs its cycle of activity. Precisely, we assume two possible cycles of activity, either $f_0 f_2 \ldots f_k$ or $f_1 f_2 \ldots f_k$. Each cycle followed by the mapping f_0 is said to be an extinction cycle and each cycle followed by f_1 is said to be an upholding cycle.

Any k-tuple $x \in \Sigma^k$ such that $\rho(x) = 1$ will be called the reset condition of a filter automaton. An activated filter automaton remains active until some reset condition coincides with a cycle of its activity.

We shall specify function ρ by the set of its reset conditions. In some k-tuples the don't care symbol "$*$" will be used; e.g. $\{03{*}2\}$ represents four reset conditions if $m = 3$.

Note that often some special classes of functions, namely the totalistic functions or even the semitotalistic functions are considered and appear useful [2, 17].

In this paper we shall consider some simplified version of the FA. We assume $\iota(\sigma) = 0$ if $\sigma = 0$, and $\iota(\sigma) = 1$ otherwise. In such a case we drop ι specifying a filter automaton. Usually we assume also that $f_1(0) \neq 0$, and that all functions f_i are one-to-one for $i \neq 0$.

For binary alphabet only four functions $f: \Sigma \to \Sigma$ exist; we denote them by **0**, A (accept), N (negate) and **1**.

A filter automaton operating over binary alphabet $\Sigma = \{0,1\}$ and specified by a program $\kappa = (f_0, f_1, \ldots, f_k)$ such that $f_0 = \mathbf{0}$ and $f_1 = \text{N}$, will be called a binary filter automaton. We shall denote the binary filter automata by $F = (\rho, \kappa)$.

We say that a filter automaton performs an idle step at the moment τ if it performs f_0 operation at that moment, otherwise it performs an active step.

Example 1. Let us consider the binary filter automaton F = ({$*00$}, (**0**,N,A,A)) with $k = 3$ and $\kappa = (f_0, f_1, f_2, f_3)$ such that $f_0 = \mathbf{0}$, f_1 = N, $f_2 = f_3$ = A. The function ρ is represented by two reset conditions. In the table below we show processing of a binary sequence.

Table 1. Processing of a binary sequence by the filter automaton F = ({$*00$}, (**0**,N,A,A)).

discrete time steps τ	...	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	...
input sequence	...	0	1	0	1	0	1	1	1	0	0	1	0	0	0	0	0	1	0	0	0	...
operations performed by F	...	f_0	f_0	f_2	f_3	f_1	f_2	f_3	f_1	f_2	f_3	f_0	f_2	f_3	f_0	f_0	f_0	f_0	f_2	f_3	f_0	...
step type (*a* - active, *i* - idle)	...	*i*	*i*	*a*	*a*	*a*	*a*	*a*	*a*	*a*	*a*	*i*	*a*	*a*	*i*	*i*	*i*	*i*	*a*	*a*	*i*	...
automaton's activity cycle #	...		←1→			←2→			←3→			←4→						←5→				...
output sequence	...	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	...

The symbols should be read from left to right in all sequences. Five cycles of activity are forced on the automaton F, however it is activated three times. The cycles #1 and #2 are upholding cycles. There are three extinction cycles: #3, #4 and #5. Note also that cycles #3 and #4 are separated by one idle step, whereas cycles #4 and #5 are separated by four such steps. We shall say that there are two series of idle internal steps (IISs).

3. Iterative processing of sequences by filter automata

An iterative process of passing a given initial configuration, a sequence a^t, through a filter automaton F is shown in Fig. 2. After each iteration over a^{t+j} the next configuration a^{t+j+1} is computed. It is easy to arrange such a computation using a pipeline-like structure of repeated filter automata F. As mentioned before, this model is equivalent to an OLIA model or a linear cellular automaton with only one (left-hand) neighbor.

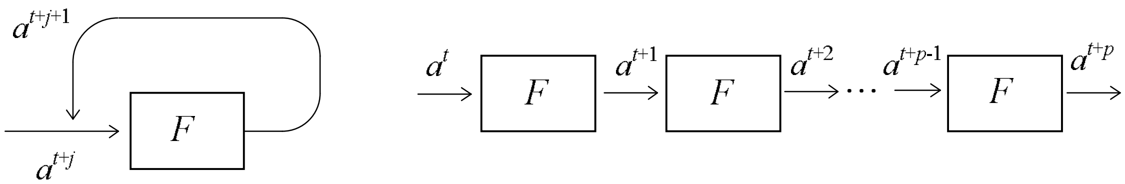

Fig. 2. Iterative processing of discrete sequences and an equivalent processing line.

We shall use the parallelograms [10] to present the results of computations iterated on filter automata. The parallelograms are understood as follows. In the first line an initial configuration a^t will be placed, and the subsequent lines will correspond to the subsequent configurations a^{t+1}, a^{t+2}, ... Sometimes, the subsequent configurations are shifted to the left by the constant value q. We will call it the range of a processing automaton. When particles are moving to the right too fast then a range $q > 0$ may be used to compensate it and make a parallelogram more readable.

A number of particular strings are observed on parallelograms. We are especially interested in periodic ones. Let us define them. Any finite sequence a^t over Σ , starting and ending with a non-zero element will be called a string. Any string a^t (passed iteratively through a filter automaton) such that $a_{i+d}^{t+p} = a_i^t$ holds for some integer p, for all $-\infty < i < \infty$, and all $t \geq 0$, is said to be a periodic string. The smallest integer p with the above property is called the period of a string. The integer d is called the displacement of a periodic string.

There is a number of terms like solitary waves, particles, solitons, basic strings, breathers, etc., used to describe various periodic strings appearing on parallelograms. Some objects, such as the root signal, were already known in the theory of signal processing [4] and in cellular automata. Using the concept of filter automata we are able to define and classify uniformly all these objects.

We define the p-periodic connected particle a^t of a filter automaton F to be a p-periodic string, such that for each $t = 0, 1, \ldots, p-1$ the automaton F is activated by a^t exactly once.

When a configuration a^t is read by a filter automaton F it may happen that F is activated only once. We say in this case that configuration a^t is connected. But it may happen as well that F is activated a number of times. Then after each extinction cycle (besides the last one) there is a series of intervals during which the automaton F remains inactive, but after which it is activated again. These intervals are called the idle internal steps (IISs). IISs allow to distinguish multiple objects occurring in complex particles. In this paper we will give only some examples of complex particles. Note also that the separation of two particles by at least one IIS indicates that they do not interact with each other at a given moment.

Example 2. Let $F = (\Sigma, \iota, \rho, \kappa) = (\{0,1,2,3\}, \iota, \{0000, 30{*}{*}, 2000\}, (f_0, f_1, f_2, f_3, f_4))$, where activating function ι and the operations of κ are given below (here f_1 is not one-to-one):

$$\iota = \begin{pmatrix} 3210 \\ 1110 \end{pmatrix}, \; f_0 = \begin{pmatrix} 3210 \\ 1300 \end{pmatrix}, \; f_1 = \begin{pmatrix} 3210 \\ 1301 \end{pmatrix}, \; f_2 = f_4 = \begin{pmatrix} 3210 \\ 3210 \end{pmatrix}, \; f_3 = \begin{pmatrix} 3210 \\ 1032 \end{pmatrix}.$$

Assume that the sequence $a^t = \ldots 01030 \ldots$ is given at the moment $t = 0$. We show below how this sequence is transformed under iterated passing through the given automaton F.

$t = 0$: . . . $0\mathbf{103}\underline{0000}00000000$ . . .
$t = 1$: . . . $0001010\underline{2000}00000$ . . .
$t = 2$: . . . $000003030\underline{2000}000$ . . .
$t = 3$: . . . $000001010\underline{3020}000$ . . .
$t = 4$: . . . $0000000\underline{3010}00000$ . . .
$t = 5$: . . . $0000000\mathbf{103}\underline{0000}00$ . . .

One can easily note a particle 103 moving to the right in the above parallelogram. Such objects exhibit soliton-like behavior during collisions. In the parallelogram we show also extinction cycles of given automaton F by underlying the associated 4-tuples of symbols.

4. Relation of a filter automaton to the PRR filtering

The parity rule recursive (PRR) filtering is determined by an update function Φ and by r - the radius of its window. These are defined as follows.

Let a^t denote a sequence $\ldots 0\, a_0^t \ldots a_i^t \ldots a_z^t\, 0 \ldots$ of elements from Σ, given at moment t, where: $0 \le t < \infty$, $-\infty < i < \infty$, $z < \infty$, and $a_0^t, a_z^t \neq 0$. According to the parity rule Φ the next configuration a^{t+1} is calculated sequentially from left to right by:

$$a_i^{t+1} = \Phi(a_{i-r}^{t+1}, a_{i-r+1}^{t+1}, \ldots, a_{i-1}^{t+1}, a_i^t, a_{i+1}^t, \ldots, a_{i+r-1}^t, a_{i+r}^t),$$

where the parity rule Φ is a totalistic [2] function $\Sigma^{2r+1} \to \Sigma$ such that

$$a_i^{t+1} = \begin{cases} 1 & \text{if } sum \text{ is even but not zero,} \\ 0 & \text{otherwise,} \end{cases}$$

where $sum = \sum_{j=-r}^{-1} a_{i+j}^{t+1} + \sum_{j=0}^{r} a_{i+j}^{t}$.

This model was called originally a parity rule filter automaton (PRFA) [15]. It was pointed out that it is equivalent to processing sequences by the window of shape given below.

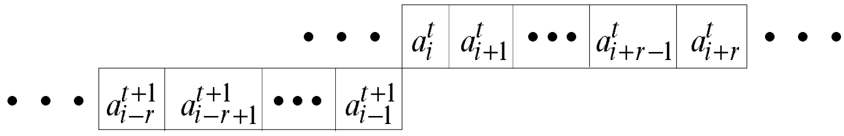

Fig. 3. Window of PRR filtering. When sliding to the right it operates as a transducer.

Theorem. The PRR filtering of radius r is equivalent to the binary filter automata of the form $F = (\{*0^r\}, (\mathbf{0},\mathrm{N},\mathrm{A},...,\mathrm{A}))$.

Proof. Let us assume that the upper part of the window of PRR filtering is shifted by r positions to the left. Let us call it an s-window. The s-window is represented by:

$$a_i^{t+1} = \Phi(\,a_{i-r}^{t+1}, a_{i-r+1}^{t+1}, ..., a_{i-1}^{t+1}, a_{i-r}^{t}, a_{i-r+1}^{t}, ..., a_{i-1}^{t}, a_i^t).$$

Because of the zero left-boundary conditions and sequential computations the updating rule with the s-window is evidently equivalent to the PRR filtering. The only difference is that each succeeding configuration is now shifted by r positions to the right compared to the results returned by original window. In other words the range of s-window is $q = 0$.

Let us check now r+2 positions of such an s-window. This is shown in Fig.4. Its first position is chosen when the leftmost non-zero element a_0^t has just entered the s-window. In Fig. 4. we omit superscripts to clarify it. By definition of Φ the function $f_0 = \mathbf{0}$ is about to be executed on a_0. Its result, $f_0(a_0) = \mathbf{0}(1) = 0$, will appear below a_0. This is step $\tau = 0$. Beginning from the moment $\tau = 1$ up to $\tau = r$, the s-window is non-empty because $a_0 \neq 0$. The parity of the sum of s-window elements is simply determined by $a = a_0 + a_\tau$, since all elements $a_1, a_2, ..., a_{\tau-1}$ appear in pairs in the s-window. If $a_\tau = 1$ the a is even. If $a_\tau = 0$ the a is odd. It means that Φ operates exactly like the functions f_2 up to f_{r+1} which are all of the type A (accept). Thus ($\mathbf{0}$, A, ..., A) constitutes the first cycle of r+1 steps of processing.

Now consider the step $\tau = r+1$. Two cases are possible. Either the elements $a_1, a_2, ..., a_r$ are all zeros or there is at least one non-zero element among them. In the first case we have the same situation as at $\tau = 0$ (or before it), so the function $f_{r+2} = f_0 = \mathbf{0}$ will be executed. The cycle may be repeated either immediately if $a_{r+1} \neq 0$, or later if another non-zero element will enter the s-window. In the second case the parity of the sum of s-window elements corresponds to the inverse of a_{r+1} so the function $f_{r+2} = f_1$ = N (negate) is about to be used. Then the cycle will go through $f_2, f_3, ..., f_{r+1}$, so it will be of the form (N, A,..., A).

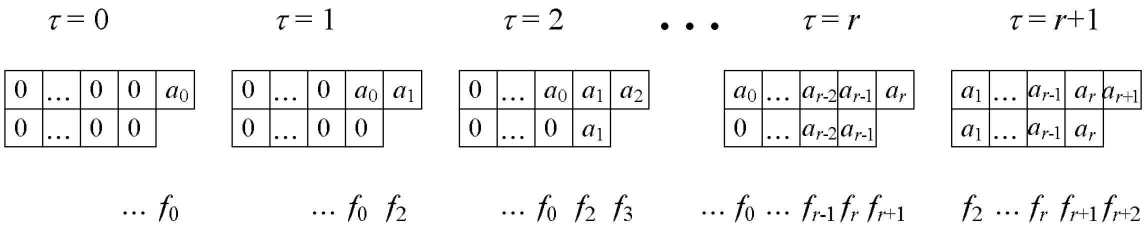

Fig. 4. Cycle of r+1 operations $f_0, f_2, ... , f_{r+1}$. At $\tau = r+1$ a new cycle may start with f_{r+2}.

We may say that the s-window is inactive if it performs operation Φ equivalent to $f_0 = \mathbf{0}$, otherwise it is active. Now it is clear that the parity rule filter automaton is a special case of the more general filter automaton model defined at the beginning of this paper. The s-window, when activated, performs the activity cycles of the form: one step of negation N followed by r steps of acceptance A. The s-window turns off its activity only if the k-tuple of the form $*0^r$ coincides with its cycle, where $k = (r+1)$. This completes the proof. **Q.E.D.**

Corollary. The filter automaton introduced above constitutes a generalization of the PRR filtering such that:

- any finite alphabet $|\Sigma| = m+1$ can be used, not only the binary one $\{0,1\}$,
- any k-tuple is allowed as the reset condition, not only the strings of the form $*0^r$, and
- any k-tuple of mappings $\Sigma \to \Sigma$ may constitute the cycle of automaton's activity; whereas in PRR filtering only two $(r+1)$-cycles are employed as the parity update rule: either $\mathbf{0}A^r$ or NA^r,
- any range q can be used on parallelograms since it is now independent of the processing automaton (in PRR filtering it is $q = r$).

Note that two other generalizations of PRR filtering have been proposed in literature [1, 6]. In [1] it is shown at first that PRR filtering is still performed if the reset condition is of the form 0^{r+1}. Then a multistate difference relation and its corresponding fast rule theorem are given. The generalization from [6] uses an arbitrary finite group instead of binary alphabet. Moreover, several two-dimensional versions of this model are also presented.

5. Particles supported by FA

Our new filter automaton model exhibits highly nontrivial and surprisingly rich particle behavior. The dynamics of its computations varies from steady patterns up to the chaos so another paper must be devoted to present it. Also a particle classification will be presented in a separate paper. Here only some examples will be given to show that new phenomena occur in the FA model. As a byproduct we show some possibilities of visualization of particles. We would like to increase awareness that the FA may appear useful in many disciplines; e.g. in signal processing, studying edge of chaos conditions, arranging pipelined computations or simulating solitonic waves.

First note that a particle may involve variable number of cycles of activity in the associated automaton which was impossible in PRR filtering. We propose to call such a particle the <u>strange particle</u> of FA. One case of a strange particle has been shown in the example 2. It is easy to verify that at $t = 4$ only one cycle of activity of F is involved, whereas during any other iteration two cycles of activity of F are forced.

The next example illustrates that an event of annihilation of particles may occur in FA. Contrary to this it was conjectured in [15] that mutual annihilations are not supported by the model of PRR filtering. Note that a totalistic reset function [2] is used in our filter automaton.

<u>Example 3.</u> Consider the filter automaton $F = (\Sigma, \iota, \rho, \kappa) = (\{0,1\}, \iota, \rho, (\mathbf{0}, N, N, A, A))$, such that $\rho^{-1}(1) = \{x \in \Sigma^4 \mid$ the number of 1's contained in x equals either zero or two$\}$. The parallelogram below shows a collision of two particles, which results in annihilation.

The left-hand particle is consistent. It is composed of four connected strings. Its period is $p = 1$. The right-hand particle is a strange particle with period $p = 2$; note that either two or three cycles of activity of F are involved. To facilitate identification of connected substrings we used the following convention: the symbols read by the automaton F during its idle steps and such that they activate F are printed in bold, and the zeros read during idle steps of F are denoted by ‘•‘.

$t = 0$: •**1**100**1**010**1**001**1**0000000••••••••**1**11110000000•••••••••••••••

$t = 1$: ••••••**1**100**1**010**1**001**1**0000000•••••**1**1010011•••••••••••••••••

$t = 2$: •••••••••••**1**100**1**010**1**001**1**0000000•••**1**11110000000••••••••••

$t = 3$: ••••••••••••••••**1**100**1**010**1**001**1**0000000**1**1010011••••••••••••

$t = 4$: •••••••••••••••••••••**1**100**1**010**1**001**1**0000011**1**1100000•••••••

$t = 5$: ••••••••••••••••••••••••••**1**100**1**010**1**001**1**1100101**1**0000000••

$t = 6$: •••••••••••••••••••••••••••••••**1**100**1**010•**1**010•**1**010•**1**100••

$t = 7$: ••••••••••••••••••••••••••••••••••••**1**100•**1**100•**1**100••••••

$t = 8$: ••

Example 4. In Fig. 5 we show a collection of colliding particles. The complex breathers [5] are among them. The associated filter automaton F is defined as follows: $F = (\Sigma, \iota, \rho, \kappa) = (\{0,1,2\}, \iota, \{0000, 1221\}, (f_0, f_1, f_2, f_3, f_4))$, where:

$$\iota = \begin{pmatrix}210\\110\end{pmatrix},\ f_0 = \begin{pmatrix}210\\000\end{pmatrix},\ f_1 = \begin{pmatrix}210\\201\end{pmatrix},\ f_2 = \begin{pmatrix}210\\012\end{pmatrix},\ f_3 = f_4 = \begin{pmatrix}210\\120\end{pmatrix}.$$

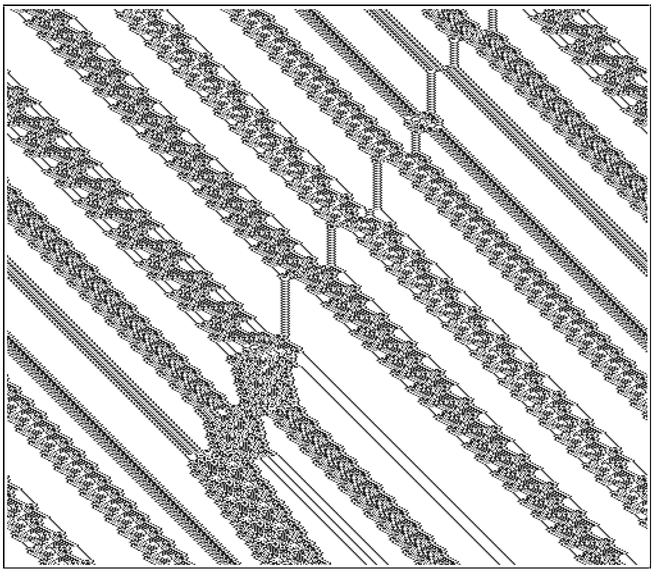

Fig. 5. Parallelogram of colliding breathers of a filter automaton; $q = 3$.

There are eight particles in the initial configuration of parallelogram in Fig. 5:

1200 1110 1122 1121 2122 1121 2121 0021 0202 2220 012,
1200 0000 1221 1220 0102 1002 1212 1221 1220 2220 012,
1200 1110 1122 1121 2121 0020 1110 1101 2,
1200 1111 0111 2211 2,
1200 1110 2010 2220 012,
1221 1222 0102 1002 1212 2112 2022 2001 2,
1221 12, and
1200 0000 1200 0000 0120 2202 2201 0210 0212 1212 2112 2022 2001 2.

Example 5. Now we present a particle of period $p = 3$ of an automaton F operating over a 5-element alphabet. Let $F = (\Sigma, \iota, \rho, \kappa) = (\{0,1,2,3,4\}, \iota, \{0000, 1000, 1100\}, (f_0, f_1, f_2, f_3, f_4))$, where the functions ι and f_i are given below.

$$\iota = \begin{pmatrix}43210\\11110\end{pmatrix},\ f_0 = \begin{pmatrix}43210\\00000\end{pmatrix},\ f_1 = \begin{pmatrix}43210\\12043\end{pmatrix},\ f_2 = \begin{pmatrix}43210\\02431\end{pmatrix},\ f_3 = \begin{pmatrix}43210\\30421\end{pmatrix},\ f_4 = \begin{pmatrix}43210\\24130\end{pmatrix}.$$

On the diagrams of Fig. 6 we assumed $q = 0$. The particle moves to the right.

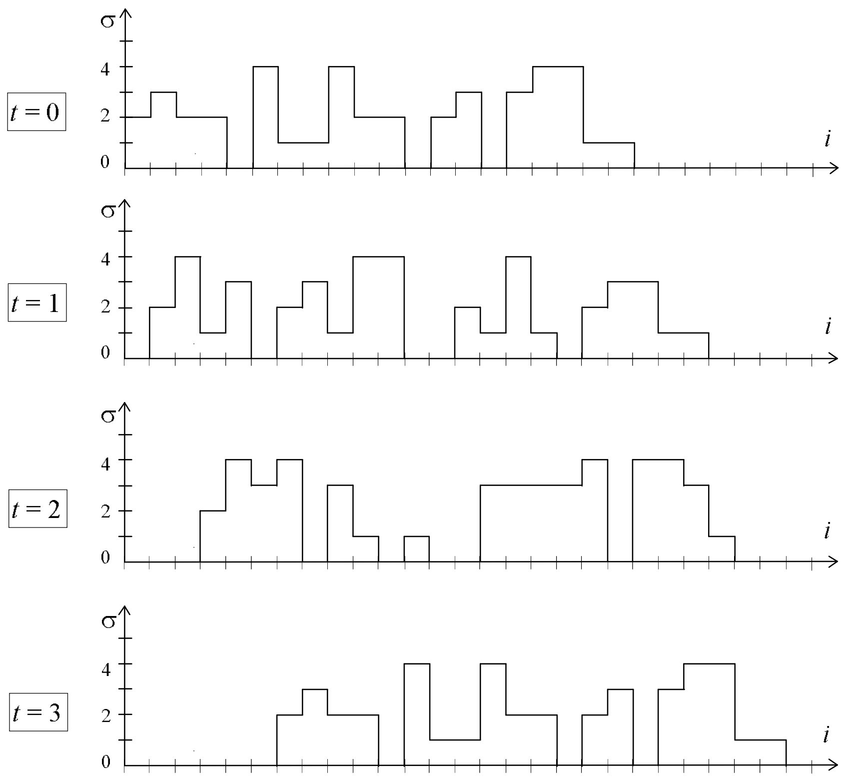

Fig. 6. Diagrams of a particle.

Example 6. In Fig.7 we show a plot of soliton-like collision of two particles 11 and 1111. The associated filter automaton is equivalent to PRR filtering with $r = 3$.

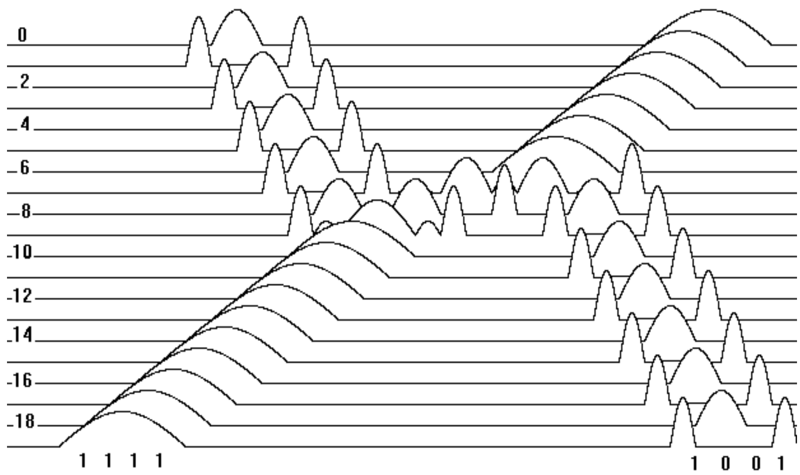

Fig. 7. Colliding particles of the binary filter automaton $F = (\{*000\}, (\mathbf{0},\mathrm{N},\mathrm{A},\mathrm{A},\mathrm{A}))$. The range is $q = 2$.

6. Conclusions

It is remarkable that the parity rule recursive (PRR) filtering of radius r, introduced by Park, Steiglitz and Thurston [15], when applied iteratively to a given binary sequence, supports solitons. The solitons were shown later [20] to be able to perform some computations. There are also another meaningful indications [3] that discrete solitons play fundamental role in complex computational systems and optical transmission [22].

In this paper we have presented a new wider class of automata, namely the filter automata. It was shown that the filter automata serve to significantly extend the PRR filtering. Any finite alphabet may be used, and any sequence of operations may transform symbols. In filter automata the extinction of processing is under control, too. Moreover the notions used to define filter automata, which are activation, operations and extinction of processing, are very natural if one would mimic a transmitting medium by a pipeline of identical automata. Thus, iterative processing of a string may be interpreted as a mutual interaction between a transmitting medium and a given signal.

Let us note here that two other extensions of PRR filtering [1] and [6] were proposed, too. In another paper we shall show how they can be covered by the filter automata ideas.

We believe that filter automata constitute a model of actual fundamental importance. Here are some arguments. Filter automata offer a natural background to implement one-dimensional, pipelined computational structures. There is a hope that in the near future they will be implemented by means of optical fibers. It is very probable that the FA are computationally universal. If so, it would have further important practical implications.

Filter automata allow to control easily their processing parameters which is crucial in looking for the edge of chaos conditions. They exhibit chaos and order when the program κ or the reset conditions $\rho^{-1}(1)$ are tuned. Let us also point out that filter automata are naturally interchangeable with programming structures because they are described by notions which correspond to the iterating loop, concatenation and conditional branching. This observation may be used to replace some programs by an equivalent sequence of appropriate (hardware) filter automata.

7. References

[1] M.J. Ablowitz, J.M. Keiser, L.A. Takhtajan. Class of stable multistate time-reversible cellular automata with rich particle content. Physical Review A, v.44, No.10, (15 Nov. 1991), pp. 6909-6912.

[2] J. Albert, K. Culik II. A simple universal cellular automaton and its one-way and totalistic version. Complex Systems, v.1 (1987) pp. 1-16.

[3] R. Das, M. Mitchell, J.P. Crutchfield. A Genetic algorithm discovers particle-based computation in cellular automata. Typescript. Submitted to the 3rd Parallel Problem-Solving From Nature Conf. March 2, 1994.

[4] J.P. Fitch, E.J. Coyle and N.C. Gallagher. Median filtering by threshold decomposition. IEEE Tr. on Acoustic, Speech and Signal Processing, v.ASSP-32, No.6, Dec. 1984, pp. 1183-1188.

[5] A.S. Fokas, E.P. Papadopoulou, Y.G. Saridakis. Soliton cellular automata. Physica D. v. 41 (1990), pp. 297-321.

[6] A.S. Fokas, E.P. Papadopoulou and Y.G. Saridakis, Coherent structures in cellular automata, Physics Letters A, v.147, No.7, (23 July 1990), pp. 369-379.

[7] A.S. Fokas, E.P. Papadopoulu, Y.G. Saridakis, M.J. Ablowitz. Interaction of simple particles in soliton cellular automata. Studies in Applied Math. v.81 (1989), pp.153-180

[8] O.H. Ibarra, T. Jiang, H. Wang. String editing on a one-way linear array of finite-state machines. IEEE Tr. on Comp. v.41, No.1, 1992, pp.112-118.

[9] K. Lindgren, M.G. Nordal. Universal computation in simple one-dimensional cellular automata. Complex Systems, v.4 (1990), pp. 299-318.

[10] P. Manneville, N. Boccara, G.Y. Vichniac, R. Bidaux (eds.). "Cellular automata and modeling of complex physical systems". Springer-Verlag, Berlin 1989.

[11] B. Mikolajczak (ed.), "Algebraic and structural automata theory". Annals of Disc. Math. v.44, Elsevier, Amsterdam, 1991.

[12] T.S. Papatheodorou, M.J. Ablowitz, Y.G. Saridakis. A rule for fast computation and analysis of soliton automata, Studies in Applied. Math. v.79 (1988), pp. 173-184.

[13] T.S. Papatheodorou, A.S. Fokas. Evolution theory, periodic particles and solitons in cellular automata. Studies in Applied Math. v.80 (1989), pp. 165-182.

[14] T.S. Papatheodorou, N.B. Tsantanis. Fast soliton automata. In "Optimal Algorithms", H. Djidjev (ed.), LNCS v. 401, Springer, Berlin 1989, pp. 41-47.

[15] J.K. Park, K. Steiglitz and W.P. Thurston. Soliton-like behavior in automata. Physica D. 19 (1986), pp. 423-432.

[16] M. Payer. Finite state machine theory as a tool for construction of systolic arrays. In "Computer Aided Systems Theory - Eurocast '89", Pichler F., Moreno-Diaz R. (eds.), Springer-Verlag, Berlin, 1990 (LNCS No. 410), pp. 212-224.

[17] A.R. Smith. Simple nontrivial self-reproducing machines. In “Artificial Life”, C.G.Langton, C.Taylor, J.D.Framer, S.Rasmussen (eds.), Addison-Wesley Publishing Comp. Reading, Massachusetts, 1992, pp. 709-725.

[18] P. Siwak. Particles of parity rule recursive filtering constructed by means of particle edge automata. Typescript pp. 1-30, 10 Jan. 1992.

[19] P. Siwak. Introduction to filter automata theory. Studia z Automatyki, T.XVIII, 1993, pp. 87-110.

[20] K. Steiglitz, I. Kamal, A. Watson. Embedding computation in one-dimensional automata by phase coding solitons. IEEE Tr. on Computers. v.C-37, No.2, 1988, pp. 138-145.

[21] H. Takahashi. The maximum invariant set of an automaton system. Information and Control. v.32, 1976, pp. 307-354.

[22] J.R.Taylor (ed.). Optical Solitons - Theory and Experiment. Cambridge Univ. Press, Cambridge, 1992.

[23] S. Wolfram. Computation theory of cellular automata. Comm. in Mathematical Physics. v.96, (1984) pp. 15-57.