Published in: Proceedings of the 2nd International Symposium on Methods and Models in Automation and Robotics. 30 August - 2 September 1995, Miedzyzdroje, Poland. S.Banka, S.Domek, Z.Emirsajlow (eds.). Printed by Wydawnictwo Uczelniane Politechnika Szczecinska. Vol. 2, pp. 467-480.

AUTOMATA AND LANGUAGES

TADEUSZ PUCHAŁKA[†], PAWEŁ SIWAK[‡]

[†] Poznan University of Technology, Institute of Electronics and Telecommunication, 60-965 Poznan, Poland, PUCHALKA@POZN1V.TUP.EDU.PL

[‡] Poznan University of Technology, Department of Automation, Robotics and Computer Science, 60-965 Poznan, Poland, psiw@agat.kari.poz.edu.pl

Abstract. The paper contains a review of basic models of automata and languages, and a presentation of some chosen problems from this domain. It reflects individual interests of each author. In the introduction a short outlook on the development of the theory of automata and languages is given. The appearance and development of some models for parallel processing, starting with the first von Neumann's cellular automata up to pipelined linear computational universal structures, are presented, too. Then some preliminaries are introduced including basic models and ideas from automata, grammars and languages theory. The main part of the paper is devoted to the idea of complementarity between automata and languages. Another part presents computational abilities of cellular automata and quite a new model of filter automata together with their particles. It shows the connections of automata with the nonlinear physics. A new solitonic phenomenon, an unstable particle just discovered, is shown for the first time at the end of the paper.

Key words. Automata, languages, grammars, cellular automata, filter automata, iterative computations, dynamical systems, discrete solitons.

1. INTRODUCTION

First, we present some motives that decided about choosing the concept of this presentation, then describe its range and the historical overview of the development of the domain which we are interested in.

1.1. The concept and range of presentation

There are two main trends in the development of the theory of automata. The first one, which historically was a little bit sooner, is connected with considering the automata to be language accepting devices. The other trend focuses the attention on their computational ability, so they are rather treated as signal processing transducers. This observation serves as a starting point in our presentation.

In the first part consisting of point 2 and 3 the basic definitions and main ideas of automata and language theory as well as automata - language dualism are presented. The second part, that is point 4, considers cellular and filter automata. It is usually emphasized that cellular automata are making use of Moore model, whereas a newly introduced class of filter automata represents transducers which follow rather the Mealy model properties. Computational capabilities of these two classes are discussed and some soliton-like dynamical processes emerging in pipes of filter automata are presented.

Because of the size limits of this paper the important problems of computational complexity of automata and languages were only mentioned in the first part. The distinguished expert in this domain, dr. Robert Knast, with whom we prepared the early sketch of this paper died prematurely last year.

The paper is closed by section 5, where the authors summarize the domain, mention its wide applications and try to predict the main lines of its further development.

1.2. A historical outline

The problems of computability were investigated already in the thirties by such mathematicians as: K. Godel, S.C. Kleene, A. Turing, E. Post. Nevertheless, in the beginning they were connected only with theoretical considerations. First references to applications associated with intensive progress in the technology of logical elements has begun only in the beginning of the fifties. At first the interests were focused on the circuits without memory, the so called combinatorial circuits. A great amount of papers have appeared treating the problems of description of switching circuits, minimization of logical expressions, functional completeness, decomposition of switching functions and technological implementations of logical circuits. Many were discussed in [12]. The synthesis and analysis problems of switching nets demanded formal description means and involved classical Boolean algebras, the threshold logic and multiple valued logical calculus.

In 1954 Mealy, in 1956 Moore and in 1959 Rabin and Scott [22] formulated the precise, simple notions and definitions for a logical circuit with memory, called an automaton. From the formal point of view the automaton is an algebraic system; additionally it makes use of the notion of state and the instances of time. These first models were deterministic and synchronous. The powerful tool in the description and research of these classes of automata became the algebras of partitions and pairs of partitions [13]. The main areas of research comprised: the analysis of extensions of automata, input-output and state realizations of automata, equivalence and other relations between automata, and state assignment problems. All these created the so called structural automata theory. Among the synchronous and deterministic automata the interests are concentrated around the classes of automata with special properties like linearity [7] or time variability [11].

Removing the postulate about a central clock from automata leads to a wide class of asynchronous models. Very special events may occur in such automata, like race phenomena, so the reliable realizations stay in quite a new light. Many papers are devoted to this class [31].

In the work of M. Rabin [21] yet another concept has been introduced, namely the probabilistic transitions of states of automata, so the deterministic model has been generalized to probabilistic machines. The continuos time and linear machines are investigated extensively.

The automata mentioned so far were associated with the strings over a finite or infinite alphabet. They were analyzed also as algebraic structures, especially as semigroups. But some trials were made also to enlarge the input objects of automata beyond the strings. As characteristic classes from this set of models the tree-automata [5] an F-automata [5, 19] must be mentioned.

In parallel to the theory of computational models with the emphasis on synthesis and analysis problems, another approach was developed starting from the analysis of natural languages. In 1959 N. Chomsky in paper [6] presented the classification of deterministic grammars. Later on various classes of precisely determined languages were identified as sets of words generable by appropriate grammars. Then generalizations were given using probabilistic productions.

Both trends, the research done in automata and that done in languages have interlaced into one formally complete entity, with classical theorems showing the mutual correspondences. On both sides of the relationship automata - languages there are complementary classifications and hierarchies. E.g. the deterministic finite automata and regular languages are related by the theorems of synthesis and analysis [15].

An important component of this research was the analysis of computational complexity. The monographs of S. Eilenberg [9] and of G. Birkhoff [3] have given the basis for future development. Various aspects of computational effectiveness have been searched. Significant results into structural complexity were introduced by S. Cook [2], J. Brzozowski and R.Knast [4].

One of the most interesting classes of automata are cellular models. The cellular automata have been introduced in 1952 by J. von Neumann and S. Ulam as formal models of auto-reproduction. This was the beginning of research the self-organizational capabilities of matter. Von Neuman showed that a configuration of a cellular net (of about 40.000 copies of simple automata) is able to reproduce itself. This was the 5n29s-type net (each automaton has 5 neighbors, including itself and 29 states). This result was later improved: in 1964 by Codd (5n8s), in 1970 by J.H. Conway (famous 9n2s-type game of life), in 1975 by Nourai and Kashef (5n4s).

Among the most interesting topics in cellular automata there are problems of their global dynamics, computational universality and simulation of processes. In the global dynamics behavior there were shown non-constructable configurations in 2-dimensional automata, the so called GOE (Garden of Eden) configurations. The looking for GOE in Conway's automaton was succeeded in 1974 (J. Hardouin-Duparc). The other well known problems are: the early bird problem, French flag problem, firing squad synchronization problem, and at last the general research in intelligent graphs [23]. Looking for cellular automata with auto-reproduction capability was not so easy. It was necessary to connect it with the notion of computational universality. A number of 2-D universal cellular automata were published. The most simple seems to be the Serizawa's model (1987) of 5n3s-type.

The most simple known since 1967 universal Turing machine is (4, 7)-machine of M. Minsky (4 tape symbols and 7 states) [16]. It was shown quite early that Turing machines can be simulated by cellular automata: A.R. Smith in 1971 showed a universal 3n18s-type cellular automaton. Basing on this, in 1987, Albert and Culik II have found an only 14 state (3n14s) computationally universal 1-D cellular automaton. This model still retains its power (becomes of 3n56s-type) even if its local function is restricted to the class of totalistic functions [1]. Recently, in 1990, Lindgren and Nordal converted the Minsky's machine to the computationally universal 1-D cellular automaton of the type 3n7s [14]. On the other side it is known that each linear cellular automaton of finite neighborhood can be converted to a pipelined structure with the nodes of Mealy-type automata. Moreover, only a dozen or so of states suffices. The conclusion is that we have the computationally universal linear structures with Mealy nodes which are simple and homogeneous and perform parallel computations in the natural way. It is clear then, that they are among the first candidates to play the role of future multiprocessing computational structures.

In the domain of simulation of processes by cellular automata there have appeared many specialized models. Homogeneous structures of deterministicly interacting "sites" are applied in the analysis of flows of liquid media, propagation of disturbances, dynamics of such complex systems as the atmosphere, populations of living organisms, or economical systems.

In the last few years, as we think, there emerges quite a new awareness of the role of the automaton model. It is connected with the idea of treating the computation as a trajectory of a certain dynamical system. In 1984, S. Wolfram has described 4 types of behavior in computations performed by binary, linear cellular automata: point, cycle, strange attractor and chaos [31]. He suggested that cellular automata from class 3 could be capable of universal computation. Since that time the computations became a new domain for physicists.

Yet another important result has appeared. In 1986, Steiglitz and others [18] described the solitonic effects observed in binary, parity rule, recursive filtering. He showed soon that solitons may be applied in performing computations [29].

Following this line we present a new class of automata, , which assumes that the computation is a result of mutual interaction between data (string being transformed) and the medium they are passed through (pipe of processors). This approach, among others, allows to describe - without using the wave equations - a fascinating physical phenomenon called the solitonic wave. We show here a new discovery in this class of events, that is an unstable particle.

The presented family of automata is called filter automata [26, 27, 28].

2. PRELIMINARIES

2.1. Classifications

It is useful for the purpose of systematic presentation, to introduce classifications concerning automata and languages. The starting point in automaton classification is to treat them as acceptors of languages of a string or non-string structure. The following levels of classification for automata - acceptors are connected with determinism or its lack, synchronism or its lack, particular properties such as linearity, variation in time, time continuity. The considerations concerning acceptors of languages of a non-string structure are limited to two classes: tree-automata and F-automata. The classification of automata is shown in Fig. 1. The classification of languages, corresponding with the former one, is depicted in Fig. 2.

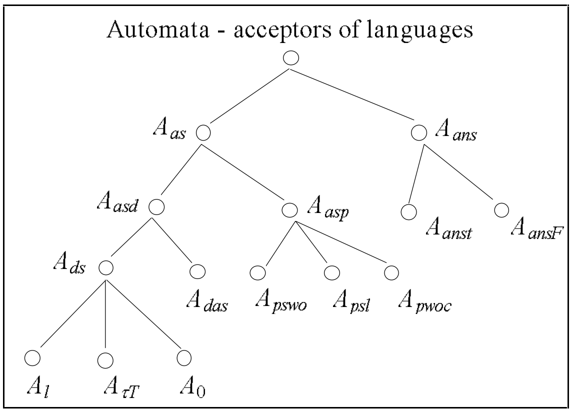

Fig. 1. Classification of automata.

It is at this point that the necessity for the definition of grammar appears. This definition is strongly connected with the language generating process.

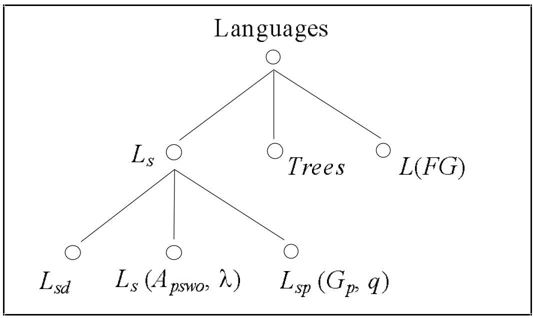

Fig. 2. Classification of languages.

2.2. Review of basic automata models

The review of basic automata models is done in correspondence to classification introduced above.

1) Automata - acceptors of string languages (A_{as}), deterministic (A_{asd}).

a) Synchronous automata.

An automaton A_{ds} with outputs and an initial state is defined as an algebraic system:

$$A_{ds} = \langle S, X, Y, \delta, \beta, s_0 \rangle, \qquad (1)$$

where: S - a non-empty, finite set of states, X - a non-empty, finite input alphabet, Y - a non-empty, finite output alphabet, δ - transition function $\delta : (S \times X)_\tau \rightarrow (S)_{\tau+1}$, β - output function $\beta : (S \times X)_\tau \rightarrow (Y)_\tau$ (Mealy automaton) or β: $(S)_\tau \rightarrow (Y)_\tau$ (Moore automaton), s_0 - an initial state, $s_0 \in S$.

Block-schema of an automaton A_{ds} is illustrated in Fig. 3. For each letter x_i read from an input string it responses with an associated output letter y_i.

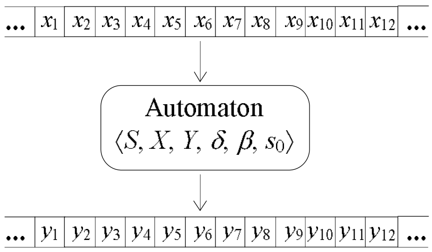

Fig. 3. Automaton A_{ds} "running" to the right.

A generalized automaton A_{gds} with outputs and an initial state is defined as a system:

$$A_{gds} = \langle S, X^*, Y^*, \delta^*, \beta^*, s_0\rangle, \tag{2}$$

where: X^* - a set of words over the alphabet X, Y^* - a set of words over the alphabet Y, δ^* - generalized transition function $\delta^*: S \times X^* \rightarrow S$ such that $\forall(s\in S)$ $[\delta^*(s, \Lambda) = s]$ and $\forall(s \in S)\forall(w \in X^+)$ $[\delta^*(s, w) = \delta(\delta^*(s, u), x)]$ where $w = ux$, $x \in X$, Λ denotes an empty word, and $X^+ = X^* \setminus \{\Lambda\}$, β^* - generalized output function which is either $\beta^* : S \times X^+ \rightarrow Y$ (for Mealy automaton) and such that $\forall(s \in S)\forall(w \in X^+)$ $[\beta^*(s, w) = \beta(\delta^*(s, u), x)]$ where $w = ux$, $x \in X$, or $\beta^* : S \times X^* \rightarrow Y$ (for Moore automaton) and such that $\forall(s \in S)\forall(w \in X^*)$ $[\beta^*(s, w) = \beta(\delta^*(s, w))]$.

A linear automaton A_l is defined as a system:

$$A_l = \langle S, X, Y, A, B, C, D\rangle, \tag{3}$$

where: A, B, C, D are zero-one matrices which satisfy the following equations:

$$[s]_{\tau+1} = A\ [s]_\tau + B\ [x]_\tau,\ \ [y]_\tau = C\ [s]_\tau + D\ [x]_\tau\,. \tag{4}$$

Assuming that $|S| = n$, $|X| = k$, $|Y| = m$, the dimensions of matrices are: A - $n \times n$, B - $n \times k$, C - $m \times n$, D - $m \times k$.

Equations (4) may be generalized by induction.

A strictly periodic, time-varying, state automaton $A_{\tau T}$ is defined as a system:

$$A_{\tau T} = \langle S, X, \delta\rangle, \tag{5}$$

where: S - is a finite sequence of finite sets of states $(S_0, \ldots, S_{T\text{-}1})$, T - a positive integer called the period of $A_{\tau T}$, δ - a sequence $(\delta_0, \ldots, \delta_{T\text{-}1})$ of transition functions δ_τ: $S_\tau \times X \rightarrow S_{\tau+1 (\mathrm{mod}\ T)}$.

Using (5) it is possible to introduce the definition of an automaton A_a - a fixed analog of $A_{\tau T}$, which is a form of generalization of the automaton $A_{\tau T}$.

An automaton A_a is defined as a system:

$$A_a = \langle S_a, X, \delta_a\rangle, \tag{6}$$

where: $S_a = S_0 \cup \ldots \cup S_{T\text{-}1}$ and δ_a: $S_a \times X \rightarrow S_a$ is such a function that:

$\forall(s \in S_a)\forall(x \in X)$ $[(s \in S_\tau) \Rightarrow \delta_a(s, x) = \delta_\tau(s, x)]$.

Among other automata (A_0), see Fig. 1, we may distinguish automata with connectivity - A_c and automata with strong connectivity - A_{sc}.

An automaton A_c is defined as a system

$$A_c = \langle S, X, Y, \delta, \beta, s_0\rangle, \tag{7}$$

where: $\forall(s \in S)\exists(w \in X^*)$ $[\delta^*(s_0, w) = s]$, and δ^* is defined as in (2).

An automaton A_{sc} is defined as a system

$$A_{sc} = \langle S, X, Y, \delta, \beta\rangle, \tag{8}$$

such that $\forall(s_1, s_2 \in S)\exists(w \in X^*)$ $[\delta^*(s_1, w) = s_2]$.

b) Asynchronous automata.

An automaton A_{das} constitutes the following algebraic system:

$$A_{das} = \langle S, X, Y, \delta, \beta, \tau_s, \tau_o\rangle, \tag{9}$$

where: τ_s - is an instant of time of executing the operation δ, τ_o - is an instant of time of executing the operation β, δ and β are such functions that $\delta(s(\tau), x(\tau)) = s(\tau + \tau_s)$ and $\beta(s(\tau), x(\tau)) = y(\tau + \tau_o)$.

From the point of view of the number of changes of an output symbol caused by a single change of an input symbol one may distinguish three classes of asynchronous automata: 1) SOC - single output change automaton; starting from an unstable state it reaches directly a stable state without a change of output, 2) MOC - multiple output change; there is a limited number of changes of output symbol, 3) UOC - unbounded output change; unlimited number of output symbol changes occurs.

2) Automata - acceptors of string languages (A_{as}), probabilistic (A_{asp}).

a) Synchronous automata without outputs (A_{pswo}).

An automaton A_{asp} is defined as a system

$$A_{pswo} = \langle S, X, \{M(x)\}, S_F, \pi_0\rangle, \tag{10}$$

where: $\{M(x)\}$ - a set of probabilistic transition matrices, $|\{M(x)\}| = |X|$, S_F - a subset of S of permissible final states, π_0 - a probabilistic initial state determined by $\pi_0 = (s_1/p_1, \ldots, s_n/p_n)$ and $p_1 + \ldots + p_n = 1$. Each element m_{ij} of matrix $M(x)$ determines the probability of transition from the state s_i to the state s_j, when an input symbol is equal to x, that is $m_{ij} = P(s_j \mid s_i, x)$ and $0 \leq m_{ij} \leq 1$, where $m_{i1} + \ldots + m_{in} = 1$. In many cases it is important to determine the probability $m(w)$ of reaching an arbitrary state belonging to S_F, by an input word $w \in X^*$. The formula is as follows:

$$m(w) = \pi_0\, M(w)\ \eta_F, \tag{11}$$

where: $w = x_1 \ldots x_k$, $M(w) = M(x_1) \ldots M(x_k)$, and

$$\eta_F = \begin{Vmatrix} \eta_F^1 \\ \ldots \\ \eta_F^n \end{Vmatrix},\ \ \eta_F^i = \begin{cases} 1, \text{ if } s_i \in S_F \\ 0, \text{ if } s_i \notin S_F \end{cases}.$$

b) Synchronous linear automaton, A_{psl}.

An automaton A_{psl} is defined as a system

$$A_{psl} = \langle \boldsymbol{S}, \boldsymbol{X}, \boldsymbol{Y}, \varphi, \psi \rangle, \quad (12)$$

where: $\boldsymbol{S}$, $\boldsymbol{X}$, $\boldsymbol{Y}$ - are n, r, m - dimensional linear spaces of vectors over field $GF(2)$, φ - is a linear probabilistic operator $\varphi : (\boldsymbol{S} \times \boldsymbol{X})_\tau \xrightarrow{pr} (\boldsymbol{S})_{\tau+1}$, such that $\sum_{s \in (S)_{\tau+1}} \varphi((\boldsymbol{s}_\tau, \boldsymbol{x}), \boldsymbol{s}_{\tau+1}) = 1$, ψ - is a probabilistic operator $\psi : (\boldsymbol{S} \times \boldsymbol{X})_\tau \xrightarrow{pr} (\boldsymbol{Y})_\tau$ which for Mealy model is such that $\psi((\boldsymbol{s}, \boldsymbol{x}), \boldsymbol{y}_1) + \ldots + \psi((\boldsymbol{s}, \boldsymbol{x}), \boldsymbol{y}_m) = 1$, and is similar for Moore model.

If an automaton is linear in the sense of the above definition, then: $\varphi(\boldsymbol{s}, \boldsymbol{x}) = A\,\boldsymbol{s} + B\,\boldsymbol{x}$, $\psi(\boldsymbol{s}, \boldsymbol{x}) = C\,\boldsymbol{s} + D\,\boldsymbol{x}$ (for Mealy automaton), (13)

where: A: $\boldsymbol{S} \xrightarrow{pr} \boldsymbol{S}$, B: $\boldsymbol{X} \xrightarrow{pr} \boldsymbol{S}$, C: $\boldsymbol{S} \xrightarrow{pr} \boldsymbol{Y}$, D: $\boldsymbol{X} \xrightarrow{pr} \boldsymbol{Y}$. The converse theorem is also true.

c) Continuous time automaton without outputs, A_{pwoc} is a generalization of automaton (10).

An automaton A_{pwoc} is defined as a system:

$$A_{pwoc} = \langle S, X, \{M(x, \tau)\}, S_F, \pi_0, T \rangle, \quad (14)$$

where: $\{M(x, \tau)\}$ - is a set of probabilistic transition matrices with each $m_{ij}(x, \tau) \in M(x, \tau)$ determining the probability of transition from the state s_i to the state s_j at moment $\tau \in T$, if an input symbol is equal to x, and $T = [0, \infty)$. It is assumed that the following conditions are fulfilled:

1) $m_{ij}(x, \tau) \geq 0$, 2) $m_{ij}(x, \tau)$ is measurable in $(0, \infty)$ and $lim_{\tau \to 0}\, m_{ii}(x, \tau) = 1$, 3) $\sum_j m_{ij}(x, \tau) = 1$, and

4) $\sum_k m_{ik}(x, \tau_1) m_{kj}(x, \tau_2) = m_{ij}(x, \tau_1 + \tau_2)$.

3) Automata - acceptors of non-string languages (A_{ans}) with a tree structure (A_{anst}).

A free structure tree automaton of a type S (sinking) over the alphabet X is defined as follows:

$$A_{ansts} = \langle Q, \delta, Q_F \rangle, \quad (15)$$

where: Q - a set of states, δ - transition function such that $\delta : Q^{[\mu]} \times X \to P(Q)$ with $\mu \geq 0$ being integer and called an index of the automaton, $Q^0 = \{\Lambda\}$, $P(Q)$ - the family of all subsets of Q, Q_F - a subset of Q of final states.

An automaton A_{ansts} is deterministic if $|\delta(Q^{[\mu]} \times X)| = 1$.

A free structure tree automaton of a type C (climbing) is defined as an algebraic system:

$$A_{anstc} = \langle S, \phi, S_F \rangle, \quad (16)$$

where: S - a set of states, ϕ - transition function such that $\phi : (S \cup \{\Lambda\}) \times X \to P(S^*)$, $\Lambda \notin S$ and $max\, lg[\phi(\{\Lambda\} \times X)] = 1$, $P(S^*)$ - a set of all subsets of S^*, S_F - a subset of final states. The number $\nu = max_{x \in X}\, lg[\phi((S \cup \{\Lambda\}) \times X)]$ is called an index of the automaton A_{anstc}. If $|\phi((S \cup \{\Lambda\}) \times X)| = 1$ then A_{anstc} is deterministic.

4) Automata - acceptors of non-string languages of F-type (A_{ansF}).

An automaton A_{ansF} is defined as a system:

$$A_{ansF} = \langle S, s_a, X, \delta, S_F, s_0 \rangle, \quad (17)$$

where: S - a finite set of states, s_a - an auxiliary state $s_a \notin S$, X - an input alphabet, δ - transition function such that $\delta : S \times X \to S \times S$, S_F - a set of final words such that $S_F \subset (S \cup \{s_a\})^+$, s_0 - an initial state $s_0 \in S$.

2.3. Grammars and languages

According to the classification as in Fig. 2, particular families of languages will be defined. Taking into account that languages are generated by grammars we should introduce basic definitions concerning the latter ones.

Deterministic grammar G_d.

A deterministic grammar G_d is a system

$$G_d = \langle V, X, P, s \rangle, \quad (18)$$

where: V - a set of non-terminal symbols, X - a set of terminal symbols, $V \cap X = \emptyset$, P - a set of ordered pairs (φ, ψ) called rules $\varphi \to \psi$, such that $\varphi \in V^+$, $\psi \in (V \cup X)^*$, s - a non-terminal initial symbol $s \in V$.

The following classification of grammars G_d has been introduced by N. Chomsky:

type 0 - each grammar is of this type,

type 1 - context grammar; $\varphi \to \psi$ are such that $|\varphi| < |\psi|$,

type 2 - free-context grammar; $\varphi \to \psi$ are such that $|\varphi| = 1$, $\varphi \in V$, $\psi \in (V \cup X)^*$,

type 3 - finite-state grammar; $\varphi \to \psi$ are of the form $A \to xB$ or $A \to x$, where: $A, B \in V$, $x \in X \cup \{\Lambda\}$.

A language is of the type i if it generates the grammar G_i of the type i ($i = 0, \ldots, 3$). A language generated by the grammar G is denoted by $L(G)$.

Probabilistic grammar G_p.

A probabilistic grammar G_p is a system

$$G_p = \langle V, X, P, \pi \rangle, \quad (19)$$

where: V - a set of non-terminal symbols, X - a set of terminal symbols; $V \cap X = \emptyset$, P - a set of probabilistic rules $\varphi \xrightarrow{pr} \psi$, π - an initial distribution of probability over V. The rule $\varphi \xrightarrow{pr} \psi$ is understood as $\varphi \to (\psi_1/p_1, \ldots, \psi_m/p_m)$, $\varphi \in V^+$, $\psi_i \in (V \cup X)^*$ ($i = 1, \ldots, m$), and $p_1 + \ldots + p_m = 1$; ψ_i is an i-th component of a sequence ψ.

Classification of the grammars G_p may be done in a similar way.

F-grammar.

To define an F-grammar we must have previously introduced concepts of forest and operations of concatenation and juxtaposition determined on the forests.

A forest over the alphabet X is a system:

$$F = \langle A', \Gamma, v \rangle, \quad (20)$$

where: A' - a finite, nonempty set of nodes, $A' = A \cup \{\Lambda \cup \varnothing\}$; Λ, $\varnothing \notin A$ are some special nodes, Γ - is a function $A' \to \underline{A}'^{*}A'^{*}$, and $v : A \to X$ with $\underline{A} = \{\underline{a} \mid a \in A\}$. The function Γ fulfills the following conditions:

a) $\forall(a \in A)[\Gamma(a) \in (\underline{A}^{*} \cup \{\Lambda\})(A^{*} \cup \{\varnothing\})]$, and $\Gamma(\Lambda) \in A^{+}$, $\Gamma(\varnothing) \in \underline{A}^{*}$,

b) $\forall(a, a' \in A)[(a \neq \varnothing) \;\&\; (\Gamma(a) = \underline{a}'w,\ w \in A^{*} \cup \{\varnothing\})$ or $(a = \varnothing) \;\&\; (\Gamma(a) = w_1\underline{a}'w_2,\ w_1, w_2 \in \underline{A}^{*}) \Rightarrow N_{a \in \Gamma(a')} = 1]$ where $N_{a \in \Gamma(a')}$ denotes a number of occurences of a symbol a in the word $\Gamma(a')$,

c) $\forall(\underline{a} \in \underline{A}')[N_{\underline{a} \in \Gamma(\varnothing)} \leq 1]$,

d) $\forall(a, a' \in A)$ if $a \in \Gamma(a')$, then $a' \notin \Gamma'^{*}(a')$ where $\Gamma'(a) = \{b \in A' \mid \Gamma(a) = w_1bw_2,\ w_1, w_2 \in (\underline{A}^{*} \cup \{\Lambda\})(A^{*} \cup \{\varnothing\})\}$, and $\Gamma'^{*}(a) = \{b \in A' \mid b \in \Gamma'(a)$ or $\exists(a_1, \ldots, a_k \in A')[a_1 \in \Gamma'(a), \ldots, a_k \in \Gamma'(a_{k-1}), b \in \Gamma'(a_k)]\}$.

The following notations are also in use:

$\underline{\Gamma}'(a) = \{b \in A' \mid \Gamma(a) = w_1\underline{b}w_2,\ w_1, w_2 \in (A^{*} \cup \{\Lambda\})(A^{*} \cup \{\varnothing\})\}$, and

$\underline{\Gamma}'^{*}(a) = \{b \in A' \mid b \in \underline{\Gamma}'(a)$ or $\exists(a_1, \ldots, a_k \in A')[a_1 \in \underline{\Gamma}'(a), \ldots, a_k \in \underline{\Gamma}'(a_{k-1}), b \in \underline{\Gamma}'(a_k)]\}$.

Each forest consisting of only one node is denoted by $\overset{x}{\downarrow}$, if $\Gamma(a) = \Lambda\varnothing$, $v(a) = x$, and similarly by $\underset{\bullet}{x}$ if $\Gamma(a) = \Lambda$, $v(a) = x$.

Let X_0^F denote a set of all forests over X. If $F_1, F_2 \in X_0^F$, $F_1 = \langle A_1', \Gamma_1, v_1\rangle$, and $F_2 = \langle A_2', \Gamma_2, v_2\rangle$ are such that $A_1' \cap A_2' = \varnothing$, then two operations on the forests may be defined: a concatenation and a juxtaposition. Assume $F = \langle A, \Gamma, v\rangle$.

A concatenation operation $F = F_1 \downarrow F_2$ is such that: $A = A_1 \cup A_2$, $v = v_1 \vee v_2$, and

$$\Gamma(a) = \begin{cases} \Gamma_1(\Lambda), & \text{if } a = \Lambda, \\ \Gamma_1(a), & \text{if } a \in A_1 \setminus \underline{\Gamma}'_1(\varnothing), \\ \Gamma_1(a)\big|\varnothing \to \Gamma_2(\Lambda), & \text{if } a \in A_1, a = \varepsilon_2(\Gamma_1(\varnothing)), \\ \Gamma_2(a)\big|\Lambda \to \varepsilon_2(\Gamma_1(\varnothing)), & \text{if } a \in \underline{\Gamma}'_2(\Lambda), \\ \Gamma_2(a), & \text{if } a \in A_2 \setminus \underline{\Gamma}'_2(\Lambda), \\ \Gamma_2(\varnothing), & \text{if } a = \varnothing, \end{cases} \tag{21}$$

where $u \mid a \to w$ means that a symbol a is replaced by a word w in a word u, and ε_2 is a projection $\Gamma_1(\varnothing)$ associated with the forest F_2.

A juxtaposition operation $F = F_1 \to F_2$ is such that: $A = A_1 \cup A_2$, $v = v_1 \vee v_2$, and

$$\Gamma(a) = \begin{cases} \Gamma_1(\Lambda)\Gamma_2(\Lambda), & \text{if } a = \Lambda, \\ \Gamma_1(a), & \text{if } a = A_1, \\ \Gamma_2(a), & \text{if } a = A_2, \\ \Gamma_1(\varnothing)\Gamma_2(\varnothing), & \text{if } a = \varnothing. \end{cases} \tag{22}$$

An F-grammar is defined as a system:

$$FG = \langle S_1^F, X_1^F, P^F, s_0\rangle, \tag{23}$$

where: S_1^F - a set of one-node forests over S, X_1^F - a set of one-node forests over X, called terminal symbols, P^F - a set of rules, s_0 - an initial forest; $s_0 \in S_1^F$, $P^F \subset S_1^F \times (S \cup X)^F$. Usually $\overset{s}{\downarrow} \Rightarrow F$ is used to denote that $(\overset{s}{\downarrow}, F) \in P^F$.

1) String languages (L_s).

a) Languages generated by deterministic grammars.

A language L_{sd} generated by a grammar G_d is a set:

$$L_{sd} = \{ w \mid w \in X^{*}, s \to w \}, \tag{24}$$

where $s \to w$ denotes that there is a sequence $(\varphi_0, \ldots, \varphi_n)$ such that $s = \varphi_0 \to \varphi_1 \to \ldots \to \varphi_n = w$.

b) Languages $L_s(A_{pswo}, \lambda)$ over probabilistic automata

A language generated over a probabilistic automaton A_{pswo}, with a cut point λ is a set:

$$L_s(A_{pswo}, \lambda) = \{w \mid w \in X^{*}, m(w) > \lambda\}, \tag{25}$$

where $0 \leq \lambda < 1$.

From (1), (10) and (25) it is seen that the deterministic automaton without outputs L_{ds} is a particular case of the probabilistic automaton A_{pswo}. Note that for an automaton A_{ds} the elements $m_{ij} \in M(x)$ are either 0 or 1, and $m(w) = 1$ if $w \in L(A_{ds})$ or $m(w) = 0$ if $w \notin L(A_{ds})$, thus $L(A_{ds}) = L(A_{pswo}, \lambda)$, for an arbitrary λ.

c) Languages generated by probabilistic grammars.

A language $L_{sp}(G_p, q)$ generated by a grammar G_p with a standard characteristic relation $q(w)$ is a set:

$$L_{sp}(G_p, q) = \{w \mid w \in X^{*}, \pi \to w, p(w) > q(w)\}, \tag{26}$$

where $p(w)$ is a probability of generating w from initial distribution π.

The relations between the families of languages $L_s(A_{pswo}, \lambda)$ and $L_{sp}(G_p^{\,i})$ are given by:

$$L_s(A_{pswo}, \lambda) \subset L_{sp}^{\,3} \subset L_{sp}^{\,2} \subset L_{sp}^{\,1} \subset L_{sp}^{\,0},$$

with assumption that $L_{sp}^{\,2}$ and $L_{sp}^{\,3}$ don't include Λ.

2) Trees.

A tree is defined to be a set $T \subset N^{*}$ such that:

$$[(t \in T) \wedge (t' \leq_l t)] \Rightarrow t' \in T, \tag{27}$$

if $t \in T$, then $\exists(n_t \in N)[[t]_T = n_t]$,

where $\leq_l$ is a lexicographical ordering of the set T, $[t]_T = |\{a \in N \mid f_n(t) \in T\}|$, $f_n: T \to T\{a\}$ is 1:1 and onto. N denotes positive integers.

An X-tree (or the tree over alphabet X) is defined to be an ordered pair $T_X = (T, v)$, such that:

T is as in (27) and $v : T \to X$. (28)

The X-tree with $v(\Lambda) = x$ is said to have an x root.

3) Languages $L(FG)$ generated by F-grammars FG.

Any set of all forests generated by an F-grammar FG is denoted by $L(FG)$ and called an F-language.

3. AUTOMATA - LANGUAGES DUALITY

At this point, using notations and definitions included in 2.2 and 2.3, the relations between the above introduced classes of automata and families of languages accepted by them will be presented.

3.1. Automata and string languages

1) Deterministic automata and regular languages.

In the following considerations the notion of a regular language is identified with a notion of a regular event. Each set $L \subseteq X^*$ is called an event over the alphabet X. If $L = \varnothing$ then such an event is called impossible. The event $L \subseteq X^*$ is represented in deterministic automaton $A_{ds} = \langle S, X, Y, \delta, \beta, s_0\rangle$ by a set of states $M \subseteq S$, if the relation $w \in L \Leftrightarrow \delta^*(s_0, w) \in M$ holds.

Two operations were defined to form an algebra of events. These are concatenation and reccurence. The concatenation $(\cdot)$ of events K and L is given by: $K \cdot L = \{w_1w_2 \mid w_1 \in K, w_2 \in L\}$. The recurrence of an event L is given by the smallest set of words consisting of all powers L^α for $\alpha = 0, 1, ...,$ with $L^0 = \{\Lambda\}$ and $L^{\alpha+1} = L^\alpha \cdot L$. The events $\varnothing$ and $\{X\}$ are elementary expressions over X.

An algebra of events (AE) is defined as a system:

$AE = \langle 0, x, +, \cdot, *\rangle$,

where: $0 \leftrightarrow \varnothing$, $x \leftrightarrow \{x\}$, + is the logic disjunction.

The inductive definition of regular expressions is:

1) 0 and x are regular expressions over X,
2) if L_1 and L_2 are regular expressions, then $L_1 + L_2$, $L_1 \cdot L_2$, L_1^* and L_2^* are also regular expressions,
3) any sequence of elements from L is a regular expression iff 1) and 2).

The value of regular expression $V(E)$ is given by:

$$V(E) = \begin{cases} \varnothing, & \text{if } E = 0, \\ \{x\}, & \text{if } E = x, \\ V(E_1) \cup V(E_2), & \text{if } E = E_1 + E_2, \\ V(E_1) \cap V(E_2), & \text{if } E = E_1 \cdot E_2, \\ [V(E_1)]^*, & \text{if } E = E_1^*. \end{cases} \qquad (29)$$

The event L over X is a regular event, if there exists such a regular expression E that $L = V(E)$.

There are two basic theorems connecting automata A_{ds} with regular events.

Theorem of analysis. Each event L represented in an automaton $A_{ds} = \langle S, X, \delta, s_0\rangle$ by a set of states $M \subseteq S$ is regular over X.

Theorem of synthesis. There exists an algorithm which for a sequence of regular events $(L_1, ..., L_k)$, $k \geq 1$, determined by a sequence of regular expressions $E_1, ..., E_k$, enables the construction of automaton $A_{ds} = \langle S, X, \delta, s_0\rangle$ in a finite number of steps; A_{ds} is such that there are the sets $M_1, ..., M_k \subseteq S$ representing the events L_i.

The proofs of both theorems may be found in [15].

These theorems are valid for finite state languages generated by deterministic finite-state grammar (see 2.3) closely related to the regular languages.

2) Probabilistic automata and languages.

It is clear from previous considerations that a deterministic automaton is a particular case of a probabilistic automaton. It leads to the conclusion that the probabilistic automata accept the regular languages.

It may be shown that the probabilistic automata accept also the non-regular languages. The example of such an automaton has been found by Moore:

$A_{pswo} = \langle\{s_1, s_2\}, \{0, 1\}, M(0), M(1), s_2, \pi_0\rangle$, where:

$$M(0) = \begin{Vmatrix} 1 & 0 \\ 1/2 & 1/2 \end{Vmatrix},\ M(1) = \begin{Vmatrix} 1/2 & 1/2 \\ 0 & 1 \end{Vmatrix},\ \pi_0 = \begin{pmatrix} s_1 & s_2 \\ 1 & 0 \end{pmatrix}.$$

The proof is based on the observation that a set of values λ (see (25)) is of continuum power, but a set of all regular languages is denumerable.

3.2. Automata and non-string languages

1) Free-structure tree automata and trees.

We say that the X-tree $T_X = (T, v)$ is accepted by an automaton A_{ansts} (see p.2) iff there exists its characteristic function $\sigma_t : T \to Q$ over the tree T_X, such that for all $t \in T$ the following conditions hold:

a) if $[t]_T = 0$, then $\sigma_t(t) \in \delta(v(t))$,

b) if $[t]_T \neq 0$, then $\sigma_t(t) \in \delta(\sigma_t(t1), ..., \sigma_t(t[t]_T), v(t))$,

c) if $t = \Lambda$, then $\sigma_t(t) \in Q_F$.

The set of all trees accepted by A_{ansts} is denoted by $L(A_{ansts})$.

Similarly, we say that the X-tree $T_X = (T, v)$ is accepted by an automaton A_{anstc} iff there exists its characteristic function $\rho_t : T \to S$ over the tree T_X, such that for all $t \in T$ we have:

a) if $t = \Lambda$, then $\rho_t(t) \in \phi(v(t))$,

b) if $[t]_T \neq 0$, then $(\rho_t(t1), ..., \rho(t[t]_T) \in \phi(\rho_t(t), v(t))$,

c) if $[t]_T = 0$, then $\phi(\rho_t(t), v(t)) \cap S_F^k \neq \varnothing$.

The set of all trees accepted by A_{anstc} is denoted by $L(A_{anstc})$.

2) F-automata (A_{ansF}) and languages accepted by them.

We assume that $X^F = X_0^F \cup \{e, \underline{e}\}$, where X_0^F - a set of all forests over the alphabet X, e - a neutral element of the monoid $\langle X_0^F, \downarrow\rangle$, $\underline{e}$ - a neutral element of the monoid $\langle X_0^F, \to\rangle$. The forest $F \in X^F$ is accepted by an automaton A_{ansF} iff $\delta(s_0, F) \cap S_F \neq \varnothing$. All the forests accepted by A_{ansF} are called to be the language $L(A_{ansF})$ accepted by A_{ansF}.

3.3. Complexity of automata and languages.

The analysis of the complexity of automata and languages is a subject of intensive research in the world. Among the most important problems there are: algebraic complexity, circuit complexity and programmable finite automata. In the first topic the fundamental results were achieved by Birkhoff [3] and by Eilenberg, particularly in relations to the varieties and pseudo-varieties [9]. The researches of circuit complexity are focused on the analysis of language classes: NC^i, AC^i, ACC. It becomes to mention here substantial results obtained by

Straubing [30], and Cook [2]. Such important problems concerning programmable finite automata like: branching programs with bounded width, the depth hierarchy of AC^0, the dot-depth hierarchy of Brzozowski [4], should be mention, too.

4. CELLULAR AND FILTER AUTOMATA

We present here two special classes of automata, namely cellular and filter automata.

Cellular automata represent a model of nets of identical processors which are usually Moore-type machines. The model is especially useful in the tasks of simulation of complex systems and performing computations by means of multiprocessing homogeneous structures. Both of these tasks are closely motivated by the development of VLSI technology.

Filter automata operate iteratively or in pipeline manner so they rather resemble the multiprocessing structures dedicated to signal processing. However, they differ from typical pipelines, e.g. systolic ones, making use of a node machine which is similar to Mealy automaton.

4.1. Cellular automata

We consider a model of synchronous cellular automata (CA).

In general a cellular automaton is understood to be a lattice, each site or 'cell' of which represents a copy of a single finite automaton. Cells interact changing their states synchronously. If they are Moore-type machines then there is a delay between any input and its resulting output. It is because in the Moore's model the output can change only if the state has been changed. It means that such CA are always stable. This is in contrast to Mealy-type CA which allow instantaneous (combinational) output from input at each cell. Each Mealy-type cell acts as a gate f_s such that $\beta(s, x) = f_s(x) = y$ for any state $s \in S$. It means that signals may propagate throughout such CA causing unstable cycles. This is why this model was considered to be not very useful [22, 30].

A cell is a Moore-type automaton $c = (S, X, \varphi)$ without outputs, where $\varphi : S \times X \rightarrow S$.

A cellular automaton is a triple $CA = (C, \nu, \varphi)$ where $C = \{c_i\}$ is a set of cells, $\nu : C \rightarrow C^n$ is a neighborhood function and $\varphi : S^n \rightarrow S$ acts as the local function of CA. It is assumed that $S \times X = S^n$ where an integer $n > 1$ determines a scope or index of neighborhood of CA. In d-dimensional CA the cells $c_i \in C$ are ordered according to their positions $i \in \boldsymbol{Z}^d$. In 1-dimensional case of CA the nearest neighbors are the most often assumed [1, 14, 29, 32] which means that $n = 3$ ($r = 1$) and $\nu(c_i) = (c_{i-1}, c_i, c_{i+1})$ for each cell c_i.

A configuration of CA is defined to be the function $f : C \rightarrow S$. S^C denotes the set of all configurations. Then, $f(\nu(c)) \in S \times X$ is the state of the neighborhood of a given cell c (including itself). A global transition function of CA is such a function $\gamma : S^C \rightarrow S^C$ that $\gamma(f) = g$ implies $\gamma(f(c_i)) = \varphi(f(\nu(c_i))) = g(c_i)$ for all $i \in \boldsymbol{Z}$.

A quiescent state $s_0 \in S$ of the CA has such a property that $\varphi(s_0, s_0, s_0) = s_0$.

We shall use the space-time diagrams (i, t) to present the results of computations performed by cellular automata or iterated on filter automata. An initial configuration a^t will be placed in its first line, and the subsequent lines will correspond to the subsequent configurations a^{t+1}, a^{t+2}, ... Index i increases to the right, and it denotes the position of a symbol x_i. Index t denotes iterations. Sometimes it is convenient to shift each subsequent configuration to the left by the constant value q. We will call it the range of a processing automaton.

Example [25]. Consider 1-D cellular automaton capable to convert binary numbers into their unary form. Let $c = (S, X, \varphi)$ be such that $S = \{0, 1, +, \times, \bullet\}$, $X = S \times S$ and $\varphi(s_{left}, s_{central}, s_{right})$ as given in table. Symbol "•" denotes the quiescent state.

Table. Local function: φ (row, old, column) = new.

0	0	1	+	×	•
0	0	0	+	+	+
1	0	0	+		+
+			+	+	
×				•	
•			•	•	

1	0	1	+	×	•
0	1	1	0		0
1	1	1	0		0
+	1	1	0		
×			×	×	
•	1	1	•		0

+	0	1	+	×	•
0	1	1	0	+	+
1	1	1	0	1	0
+			×	×	+
×				×	+
•			1	1	

×	0	1	+	×	•
0			+	+	•
1			1	1	
+			×	×	+
×	•	×	×	×	×
•		×		×	×

•	0	1	+	×	•
0					+
1					+
+					×
×				•	•
•	•	•	×	•	•

We show below how the conversion proceeds in CA.

```
t= 0 .1....10.....11.....100.......
t= 1 .0+...1++....10+....10++......
t= 2 ..+x...0+x...1++x...1+0+x.....
t= 3 .x1+....++....0x+....1+++.....
t= 4 .xx0x..x1+x....++x....0x+x....
t= 5 .x.....xx1+...x1x+.....+x+....
t= 6 .x.....xxx0x..xx1+x...x1x+x...
t= 7 .x.....xx.....xxx1+...xx1x+...
t= 8 .x.....xx.....xxxx0x..xxx1+x..
t= 9 .x.....xx.....xxx.....xxxx1+..
t=10 .x.....xx.....xxx.....xxxxx0x.
t=11 .x.....xx.....xxx.....xxxx....
t=12 .x.....xx.....xxx.....xxxx....
```

One of the most important results in cellular automata says that they are able to simulate any universal Turing machine $TM = (S, T, \delta, \beta, \rho)$; $\rho : S \times T \rightarrow \{L, R, H\}$ determines the moves of the machine's head, T - denotes a set of tape symbols.

For any universal TM one may use the $CA = (C, \nu, \varphi)$ with $\nu(c_i) = (c_{i-1}, c_i, c_{i+1})$ and the local function φ as proposed by Smith [24], and in [14]:

$\ldots x_0 x_1 \mathbf{s}{\rightarrow}\mathbf{x_2} x_3 \ldots$ $\qquad$ $\ldots x_0 x_1 \mathbf{s}{\rightarrow}\mathbf{x_2} x_3 \ldots$
$\ldots x_0 x_1 \mathbf{y_2}\mathbf{t}{\rightarrow} x_3 \ldots$ $\qquad$ $\ldots x_0 x_1 {\leftarrow}\mathbf{t}\mathbf{y_2} x_3 \ldots$

$\ldots x_0 \mathbf{x_1}{\leftarrow}\mathbf{s} x_2 x_3 \ldots$ $\qquad$ $\ldots x_0 \mathbf{x_1}{\leftarrow}\mathbf{s} x_2 x_3 \ldots$
$\ldots x_0 {\leftarrow}\mathbf{t}\mathbf{y_1} x_2 x_3 \ldots$ $\qquad$ $\ldots x_0 \mathbf{y_1}\mathbf{t}{\rightarrow} x_2 x_3 \ldots$

The state set S' of each cell $c = (S', X, \varphi)$ consists of all tape symbols T together with $\{(\leftarrow s), (s\rightarrow)\}$ - all combinations of left/right moves and internal states $s \in S$ of the Turing machine.

Note also that any 1-D cellular automaton may be simulated by a linear pipelined structure of Mealy-type automata. The following construction may be used: assume the Mealy-type automaton $A = (S, T, T, \delta, \beta)$ to be such that if $\varphi(x_1, x_2, x_3) = x_2'$ holds for $CA = (C, \nu, \varphi)$ where $c = (T, X, \varphi)$ then $S = T \times T$, $\delta((x_1, x_2), x_3) = (x_2, x_3)$ and $\beta(x_1, x_2, x_3) = \varphi(x_1, x_2, x_3) = x_2'$.

Example. Consider the Mealy machine given below with S = {0, 1, ..., F} and T = { .,1,0,+,×}. Its space-time diagram shows the conversion of binary number 5 into its unary form. Note that output strings are shifted to the right when compared with the diagram shown earlier for 1-D CA. A range q = 1 could be used to compensate it.

	δ	β
	.10+×	.10+×
0	01234	...×.
1	56789	011..
2	567EC	.....
3	A67DB	...11
4	0F2D4	××.××
5	01234	+....
6	56789	0110.
7	567EC	+00++
8	A67DB	01101
9	0F2D4	...11
A	01234	×....
B	0F2D4	+..××
C	0F2D4	...++
D	A67DB	+..××
E	A67DB	+110+
F	56789	...××

t	iterated strings
0	.101...................
1	..100+.................
2	...10++×...............
3	....1+0×+..............
4	.....1+++×.............
5	.......0××+............
6	.........+×+×..........
7	.........×1××+.........
8	..........××1×+×.......
9	...........×××1×+......
10	............××××1+×....
11	.............×××××1+....
12	..............××××××0×..
13	...............×××××....
14	................×××××...
15	.................×××××..
16	..................×××××.

There is a vast number of other specific models of cellular automata and their applications. Let us mention CML (coupled map lattices), PDE (Partial Difference Equations), LGA (Lattice-Gas Automata) [32]. Cellular automata were considered as parallel processing computers, as a test-bed for genetic algorithms [17] and in modeling of physical [8], biological [24] and many, many other phenomena.

4.2. Filter automata

In this section we shall briefly review some of the most important results associated with filter automata (FA) [26, 27, 28].

We assume that filter automata operate as sequential transducers. They transform a given string a^t iteratively which leads us to the pipelined linear computational structure shown in Fig. 4.

$$\xrightarrow{a^t} \boxed{F} \xrightarrow{a^{t+1}} \boxed{F} \rightarrow \cdots \xrightarrow{a^{t+p-1}} \boxed{F} \xrightarrow{a^{t+p}}$$

Fig. 4. A linear pipelined computational structure.

The analysis of computational process performed in such a line reveals its similarity to the propagation of disturbances in excitable media. The behavior typical for dynamical systems like point, cycle, strange attractor and deterministic chaos may be easily observed [32, 33].

At each step of iterative computation the string a^{t+j} is converted into a^{t+j+1} by a successive copy of automaton F. Such pipelined computational structures are known also as the one-way linear cellular automaton or one-way linear iterative automaton (OLIA model) [1, 14, 24, 28]. Such a pipe becomes the natural model of the linear transmitting medium (e.g. optical fiber) but only if some particular automaton is applied as its node F. Filter automata may play such a role.

On the space-time diagrams of iterative computations of filter automata the very special substrings which are called particles are visible. The particles are periodic, they move with various speeds, and they are able to collide in a non-destructive manner. The interactions of particles bear a rather close resemblance to solitonic waves which are well known in nonlinear transmitting media. Solitons are remarkably stable under disturbances so they are considered to be especially useful in transmission. Colliding solitons do not lose information, they only transform it. One may then speculate that future computations could be performed during the transmission of a signal throughout special optical fibers which will operate as the particular linearly extended multiprocessor structure. This is why we consider the filter automata which are capable of supporting soliton-like particles to be so important.

Filter automata constitute a new, well defined, special class of sequential transducers [26, 27, 28]. They feature the following notions:

(1) The states may be eliminated; instead two modes of operation are in use: a filter automaton is either inactive (extinguished or idle) or active.

(2) Three notions describe how it works:
- activation (conditions which force its activity),
- processing (operations which convert read symbols), and
- extinction (conditions which cause the automaton to be idle).

(3) The activity of filter automata is a cyclic process; a sequence of operations (program) is performed one at each instant τ of time. The cycle of operations can be shortened, and run again or extinguished. A number of distinct cycles is possible.

(4) The responses of filter automata are available by the same principle as in Mealy machine from Fig. 3 where at each moment τ - when its state s is fixed - each output symbol y_τ depends on the input letter x_τ that is read by the machine at this moment; so it performs operation $f_s : X \rightarrow Y$.

(5) The inactive mode is stable, which means that there is an input symbol x_0 (so called quiescent or zero symbol) which rests a filter automaton in inactive mode until an activating symbol is read. The output returned by the inactive automaton is also assumed to be quiescent.

It seems that such a family of automata, which are cyclic and may form stable spatial patterns of signals catches crucial features necessary to support soliton-like particles.

The phenomenon of a particle and its solitonic collision was reported for the first time in [18] in the context of the one-dimensional, binary, parity-rule recursive filtering (PRR filtering). Later in [29] the particles of PRR filtering were even shown to be able to perform meaningful computations.

The PRR filtering was analyzed by Ablowitz, and then generalized in [28] and also, independently in [10] by Fokas et al., where the algebraic groups were involved. We refer to the model given in [10] as to the FPS filtering. The quite recent generalizations were proposed in [27]. They cover the models of PRR and FPS filtering, and are symptomatic with features (**1**) - (**5**), mentioned above.

Let us present precisely the ideas which stand behind filter automata starting with FPS filtering as introduced in [10]. Let $X = \{0, x_1, ..., x_n\}$ be a finite group with its associated operation denoted by $\otimes$. Let 0 be the identity element, so there holds $0 \otimes x = x \otimes 0 = x$. Let $\bar{x}$ be the inverse of x, i.e. $\bar{x} \otimes x = x \otimes \bar{x} = 0$. For X being any finite group let us take a sequence a^t at the moment t : $...0\, x_1^t \,...\, x_i^t \,...\, x_l^t\, 0...$, where $l < \infty$ and $x_1^t \neq 0$, $x_l^t \neq 0$. Then a sequence at the moment t+1 is determined by:

$$x_i^{t+1} = \begin{cases} 0, \text{ if all factors } x_i \text{ of } S_i \text{ are } 0, \text{ or} \\ S_i \otimes z \otimes \bar{a}_{i+r}^t, \text{ otherwise} \end{cases}$$

where:

$$S_i = \sum_{j=i}^{i+r-1} \otimes (x_{j-r}^{t+1} \otimes \bar{x}_j^t) \text{ and}$$

$$z := \begin{cases} x_{i+r}^t, \text{ if all factors } x_i \text{ of } S_i \text{ are } 0, \text{ or} \\ z, \text{ otherwise} \end{cases},$$

and it is assumed that for $i \to -\infty$ all factors x_i of S_i are all zero.

In the case of Abelian group (Z_n, $\oplus$), where $Z_n = \{0, 1, ..., n\text{-}1\}$, and the symbol $\oplus$ denotes the modulus n addition, the above definition is reduced to the classical PRR filtering [10, 18].

The idea of FPS filtering is to generalize PRR filtering: any finite group is allowed instead of the binary alphabet X, and a new memory variable denoted by z is introduced. Further generalizations lead us to the family of filter automata [27]. To show the essence of filter automata we quote the definition of a general model.

A cycle-variable extended filter automaton with memory (briefly CEFAM) is a formal model of processing sequences over the finite alphabet X. We denote it by $F = (X, \iota, \mu, \kappa)$, and define as follows:

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

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

$\mu : X^{\leq k} \to X$ is a partial mapping called the control function of F, with $k > 1$,

$\kappa = (h_0, h_1, ..., h_m, f_1, ..., f_r)$ is a sequence of mappings $X \to X$, called the program of F ($r = k$-1), which generates outputs in discrete time steps τ according to the following order:

– $\text{next}(h_0(x_\tau)) = \begin{cases} h_0(x_{\tau+1}) \text{ if } \iota(x_\tau) = 0 \\ f_1(x_{\tau+1}) \text{ if } \iota(x_\tau) \neq 0 \end{cases}$, and $z := x_\tau$

– $\text{next}(h_i(x_\tau)) = \begin{cases} h_j(x_{\tau+1}) \text{ if } \mu(x_\tau) \text{ is defined} \\ f_1(x_{\tau+1}) \text{ otherwise} \end{cases}$,

where $j = \mu(x_\tau)$ and $0 < i \leq m$,

– $\text{next}(f_i(x_\tau)) = \begin{cases} h_j(x_{\tau+1}) \text{ if } \mu(x_{i \div \tau}) \text{ is defined} \\ f_{i+1}(x_{\tau+1}) \text{ otherwise} \end{cases}$,

where $j = \mu(x_{i \div \tau})$, and $0 < i < r$,

– $\text{next}(f_r(x_\tau)) = \begin{cases} h_j(x_{\tau+1}) \text{ if } \mu(x_{r \div \tau}) \text{ is defined} \\ h_z(x_{\tau+1}) \text{ otherwise} \end{cases}$,

where $j = \mu(x_{r \div \tau})$.

The string $x_{i \div \tau} = (x_{\tau-i}, x_{\tau-i+1}, ..., x_\tau)$ denotes an (i+1)-segment which is processed within the current cycle of automaton's activity.

Note that only three notions are involved: activation (ι), operations (κ) and extinction (μ) of processing. We believe, and it is observed in experiments, that these notions correspond well to the interaction between a transmitting medium and data when they are passed through it.

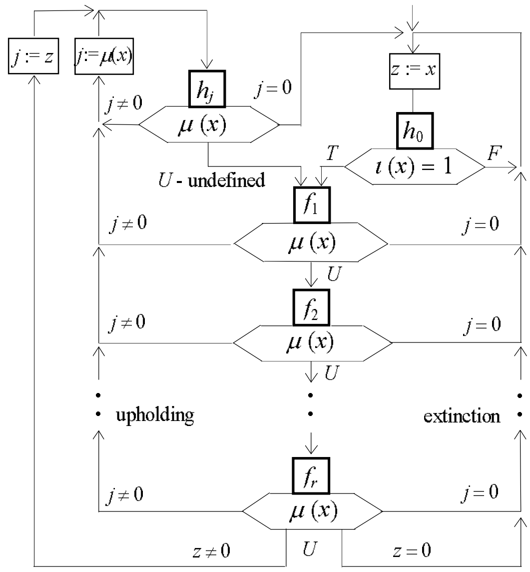

Fig. 5. A flow chart of processing performed by F (symbol $x \in X^{\leq k}$ denotes the current segment $x_{i \div \tau}$).

When the filter automaton is activated, successive operations of its program κ are executed; we say that the automaton performs its cycle of activity. Precisely, the possible cycles are $(h_i, f_1, ..., f_l)$, where $l \le r$. Each cycle followed by h_0 is said to be an extinction cycle and each cycle followed by h_i with $i \ne 0$ - an upholding cycle. We say that the filter automaton performs an idle step at the moment τ if it performs h_0 operation at that moment.

Now, let us present the particles of filter automata. Assume $X = \{0, 1, ..., m\}$. Any finite sequence a over X starting and ending with a nonzero element will be called a string. If string a^t (with enough zeros before and after) passed iteratively through a filter automaton is such that $a_{i-d}^{t+p} = a_i^t$ holds for some integer p, for all $-\infty \le i \le +\infty$, and all passes $t \ge 0$, then it is said to be a periodic string. The smallest integer p with the above property is called a period of a string. The integer d is called the displacement of a periodic string. The ratio d/p determines how the speed of a moving, periodic string has been slowed down (if $d = 0$ then the particle's speed is maximal).

In general it is not easy to define precisely the solitons. Similarly in discrete models, it is often difficult to decide whether the two substrings should be classified as a multiple or a single particle. Literature about the subject leaves this problem open or does not answer questions about it precisely [10, 26]. Let us start with a connected (or coherent) particle.

The p-periodic connected particle of a filter automaton F is a p-periodic string a^t such that for each $t = 0, 1, ..., p$-1 the automaton F is activated by a^t exactly once. If there are multiple activations of the associated filter automaton then a number of possibilities emerge, and various families of particles may be distinguished [26]. Shortly, the filter automata serve as classifiers of particles.

Yet also another point is important. Any specific particle of a filter automaton F may be described completely by its left edge symbols $\varepsilon = (x_1^{p-1}, x_1^{p-2}, ..., x_1^0)$ and a sequence $\Delta = (d_{p-1}, d_{p-2}, ..., d_0)$ of particle "row" displacements; $d = d_{p-1} + ... + d_0$.

We show below a space time diagram of a particle of a certain filter automaton F over $X = \{0, 1, 2, 3\}$, equivalent to FPS filtering [10, 27]; $d = 12$, $p = 9$.

```
t=0:  ··10231101----··············  d0 = 1
t=1:  ····23010130----············  d1 = 2
t=2:  ·····30121302----···········  d2 = 1
t=3:  ·······12230210----·········  d3 = 2
t=4:  ········22332103----········  d4 = 1
t=5:  ·········23301032----·······  d5 = 1
t=6:  ··········33030322----······  d6 = 1
t=7:  ···········30313221----·····  d7 = 1
t=8:  ·············31022110----···  d8 = 2
t=9:  ··············10231101----··  d9 = 1
```

Fig. 6. Particle of some FA, with $\Delta = (d_8, ..., d_0) = (2,1,1,1,1,2,1,2,1)$ and $\varepsilon = (3,3,3,2,2,1,3,2,1)$.

In Fig. 7 we show a fundamental phenomenon of solitonic collision of two particles of a simple filter automaton.

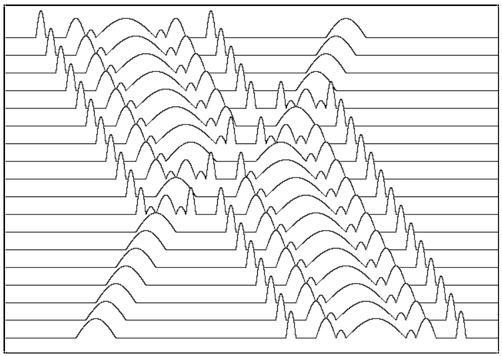

Fig. 7. Solitonic collision of two particles of F.

The solitonic collision means something special. First, when the particles approach closely enough they attract one another, just like nuclear particles do. Then, they interact and later reappear again and travel separately. Each particle precisely regains its former identity. So their shapes, speeds, amplitudes and internal frequencies are preserved under collision. The only memory of the collision is a translational shift. Note that the spatial positions of particles are changed a little, just as if they were attracted in the first phase of collision. In general it is also required that such a nonlinear superposition of n particles must occur for an arbitrary n. At the end an example of some complex collision is given.

The fundamental result from filter automata theory says that space-time diagrams of particles may be generated by a ring net of processors F^1 associated with a given filter automaton $F = (X, \iota, \mu, \kappa)$, Such a multiprocessing computation is characterized by:

- instructions (represented by the program κ),
- inter processor delays $\Delta = (d_{p-1}, d_{p-2}, ..., d_0)$,
- data (represented by the particle edge symbols $\varepsilon = (x_1^{p-1}, x_1^{p-2}, ..., x_1^0)$), and
- synchronization (determined by the function μ).

Having any three out of these four elements one may look for the remaining one.

At the end, quite a new discovery in the behavior of filter automata particles is presented for the first time. This is a fusion of two particles into a "heavier" unstable one which is able to collide solitonically. The unstable particle is such that decays after some pseudo-periodic existence.

Example. Consider the filter automaton F. It is given by: the alphabet $X = \{0, 1, 2, 3\}$, the activating function ι such that $\iota(x) = 1$ for all $x \ne 0$, and the only extinction determined by $\mu\,(0000) = 0$. Its program $\kappa = (h_0, h_1, h_2, h_3, f_1, f_2, f_3)$ is as follows:

$$h_0 = \begin{pmatrix} 3210 \\ 0000 \end{pmatrix},\; h_1 = \begin{pmatrix} 3210 \\ 2203 \end{pmatrix},\; h_2 = \begin{pmatrix} 3210 \\ 1032 \end{pmatrix},$$

$$h_3 = \begin{pmatrix} 3210 \\ 0321 \end{pmatrix},\; f_1 = f_3 = \begin{pmatrix} 3210 \\ 3210 \end{pmatrix},\; f_2 = \begin{pmatrix} 3210 \\ 2310 \end{pmatrix}.$$

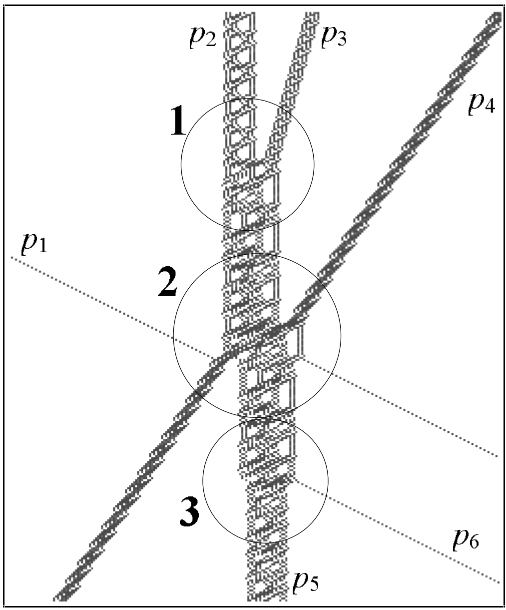

Fig. 8. Four interacting particles (p_1, p_2, p_3, p_4).

In Fig. 8 the four interacting particles are shown. Three entirely new phenomena are visible in areas 1, 2, and 3:

1) two particles (p_2, p_3) hold together and form an unstable particle (its lifetime is 160 iterations),
2) two additional particles (p_1, p_4) collide with the unstable particle without any destruction, and
3) the life of the unstable particle is over; it decays onto two particles (p_5, p_6).

The fundamental question whether (and if so, then to what extent) the model of filter automata is capable to describe physical reality of solitonic waves is open.

5. FINAL REMARKS

1. The principal question asked usually about automata is: what are they and what are they used for ? The answer is that they are formal abstract models of computational processes and may be applied in arranging computations. Their most simple versions operate over strings. They have found application in the following tasks:

a) classifying sequences (languages, recognition of 1-D objects, DNA sequencing, lexical analysis, parsers, ...).
b) string generation (grammars, developmental systems, artificial life, ...),
c) string processing (conversion, filtering, extrapolation, compression, cryptography, ...),
d) simulation of complex systems (complex nets, cellular nets, multiprocessing structures, discrete grain media, ...),
e) control (software and hardware digital technology, sequenters, arbiters, designing tools, management, ...),
f) performing computations (processors, computer programs, computability, efficiency of computations, ...),
g) simulation of processes (computations seen as trajectories of a dynamical system in generalized phase space, propagation of disturbances, flow of fluids, percolation, ...).

The tasks from a) and b) utilize mainly the linguistic aspects of automata, whereas those from c) and d) are much closer to automata treated as computational transducers. Group e) is strictly connected with digital technology and logical circuits. Applications f) result from theoretical computer science and make use of the analysis of algorithms, computability, computational universality and the computational costs. Last but not least the approach g) reflects the fact that any computation is completed on the basis of some particular physical process. The phase portrait of this associated process determines uniquely the possible results of computation.

2. The domain covered traditionally by automata and languages theory became very wide and the dynamics of its progress is all the time significant. As a result, for the purpose of this presentation we had to decide to choose only some fundamental classes and representative problems. Nevertheless, we have tried to show the automaton as an accepting device as well as a computational transducer.

3. Because of the time and space limits in this work many important models and facts were omitted. For example such models like the Turing machine, Post machine, Ginsburg machine, (α, k)-machine as well as many other important problems relevant to the complexity of automata and languages were not even mentioned. Another relevant presentation, showing the need to capture the distinct types of symbolic information processing mechanisms, has been given in [8] from the point of view of physics.

4. The state of literature supports the conclusion that we may still expect further development of the theory of automata and languages and of their complexity. Some trials to characterize the computational limits and usefulness of models which operate on non-string data are very likely to be continued. Also, it seems to be sure that further progress in pipelined homogeneous and cellular structures should provide effective fine grain means to perform in practice massive parallel computations. Then, one may expect that the analysis of complexity and the programmability of such structures will become relevant and important. Let us call this trend the classical development of automata theory.

5. The results of recent years reveal more and more connections between automata models and physics of computations [33]. The computation is a physical process so the limitations imposed by the laws of physics on the processing of information cannot be

omitted. It is accepted in physics that computations could be performed reversibly, so there was a need to look for the models that could be reversible. It has appeared that approximately reversible processors can be realized in practice (K. Morita).

It also follows that the right computational model may be crucial in better understanding of some simulated processes. Pipelined filter automata as the models of transmitting media seem to be a perfect example. Solitonic phenomena should be tractable much easier with FA than with the widely used nonlinear Schrödinger equations. We suspect that the role of automata theory, especially of cellular models, will increase both in classical and quantum physics. Let us call this trend the new, non-classical development of automata theory.

6. REFERENCES

1. Albert J., Culik II K.: A simple universal cellular automaton and its one-way and totalistic version. Complex Systems, v.1 (1987), pp. 1-16.
2. Beame P., Cook S., Hoover J.: Log-depth circuits for division and related problems, SIAM J. Computing, vol.15, 1986, pp. 994-1003.
3. Birkhoff G.: Lattice theory, 2nd ed., AMS Colloquium Publications, vol.25, 1948.
4. Brzozowski J., Knast R.: The dot-depth hierarchy of star-free languages is infinite, JCSS, vol. 16, 1978, pp. 37-55.
5. Buchi J.R.: Weak second-order arithmetic and finite automata, Math. Log. Grund. of Math. 6, 1960, pp. 66-92.
6. Chomsky N.: On certain formal properties of grammars, Information and Control, No. 2, 1959, pp. 137-167.
7. Cohn M.: Properties of linear machines, Journal of ACM, No. 11, 1965.
8. Crutchfield J.P.: The calculi of emergence: computation, dynamics and induction. Physica D, v.75, 1994, pp. 11-56.
9. Eilenberg S.:Automata, languages and machines, vol. A, B, Acad. Press, New York, 1974, 1976.
10. Fokas A.S., Papadopoulou E.P., Saridakis Y.G.: Coherent structures in cellular automata, Physics Letters A, vol.147, No. 7, 23 July 1990, pp. 369-379.
11. Gill A.: Linear sequential circuits, McGraw-Hill Book Company, New York, 1967.
12. Harrison M.A.: Introduction to switching and automata theory, McGraw-Hill Book Company, New York, 1965.
13. Hartmanis J., Stearns R.E.: Algebraic structure theory of sequential machines, Prentice Hall Inc. Englewood Cliffs, New York, 1966.
14. Lindgren K., Nordal M.G.: Universal computation in simple one-dimensional cellular automata. Complex Systems, v.4 (1990), pp. 299-318.
15. McNaughton R., Yamada H.: Regular expressions and state graphs for automata, IRE Transactions on EC, vol.EC-9, March 1960, pp. 39-47.
16. Minsky M.: Computation: finite and infinite machines, Prentice-Hall, Inc., Englewood Cliffs, N.J., 1967.
17. Mitchell M., Crutchfield J.P., Hraber P.T.: Evolving cellular automata to perform computations: mechanisms and impediments. Physica D, v.75 (1994), pp. 361-391.
18. Park J.K, Steiglitz K., Thurston W.P.: Soliton-like behavior in automata. Physica D. v.19, 1986, pp. 423-432.
19. Puchałka T., Foryś M., Foryś W.: F-automata mathematical models as the representation of F-languages, 1st World Conference on Mathematics at the Service of Man, Universidad Politecnica de Barcelona, Preprints of session 2/2, 1977, pp. 21-22.
20. Puchałka T.: Theory of switching systems, vol.1-3 (in polish), Polish Academy of Sciences, Branch Poznan, Poznan, 1976-1980.
21. Rabin M.O.: Probabilistic automata, Information and Control, v. 6, No.3, 1963, pp. 230-245.
22. Rabin M.O., Scott D.: Finite automata and their decision problems, IBM Journal Res. Dev., vol. 3, No.2, 1959, pp. 114-125.
23. Rosenstiehl P., Fiksel J.R., Holliger A.: Intelligent graphs: networks of finite automata capable of solving graph problems, in "Graph Theory and computing" (ed. R.C.Read), New York, Acad. Press, 1973, pp. 219-265.
24. Smith A.R.: Simple nontrivial self-reproducing machines, in “Artificial Life”, C.G.Langton, C.Taylor, J.D.Farmer, S.Rasmussen (eds.), Addison-Wesley Publishing Comp. Reading, Massachusetts, 1992, pp. 709-725.
25. Siwak P.: Minimal state cellular segment generation, 13th World Congress on Comp. & Applied Mathematics, July 22-26, 1991, Dublin. IMACS'91 Proceedings, v.2, pp.714-715.
26. Siwak P.: Introduction to filter automata theory, Studia z Automatyki, v.XVIII, 1993, pp. 87-110.
27. Siwak P.: Models of filter automata, (in polish) Sudia z Informatyki i Automatyki, T.XIX, 1994, pp. 77-97.
28. Siwak P.: Filter automata and PRR filtering. Journal of Technical Physics, No.2, 1995. pp. 153-163.
29. Steiglitz K., Kamal I., Watson A.: Embedding computation in one-dimensional automata by phase coding solitons. IEEE Tr. on Computers, vol. C-37, No.2, 1988, pp. 138-145.
30. Straubing H.: Finite automata, formal logic and circuit complexity, Birkhauser, Boston, Basel, Berlin, 1994.
31. Unger S.H.: Asynchronous sequential switching circuits, J.Wiley and Sons, New York, 1989.
32. Wolfram S.: Cellular Automata and Complexity Collected Papers, Addison-Wesley Publishing Comp., Reading, Massachusetts, 1994.

33. Zurek W. (ed.): Complexity, Entropy and the Physics of Information. Vol.VIII Santa Fe Institute Studies in the Science of Complexity. Addison-Wesley Publ. Comp. Reading, Massachusetts, 1990.