Automata-Based Anticipatory Systems[1]

Volodymyr.G. SKOBELEV
Institute of Applied Mathematics and Mechanics
of National Academy of Sciences of Ukraine,
Rose Luxemburg Str., 74, Donetsk, 83114, Ukraine
Email: skbv@iamm.ac.donetsk.ua

Abstract
The given paper investigates some strong anticipation characteristics, inherent to Automata Theory Problems. It is extracted anticipation's role in linear automaton's Controllability/Observability analysis. Via decision-making process presentation in terms of some special walks on some directed labelled multigraph, There is characterized strong anticipation for Problems of weakly initialized finite automaton's internal states identification, as well as of maximal supervisor's design for any discrete event automata-based system. Presentation of winning strategy's design for any Two-Players Game on a graph in terms of design of multi-headed Turing Machine with some arbiter and independently controlled heads outlines some general anticipatory characteristics, inherent to distributed computing.
Keywords: anticipation, automata, identification, control, algorithms on graphs

1 Introduction

Nowadays, a variety of discrete models, weakly interacted each with the others, are used in Computer Science and its applications extensively. Although the basic principles of these models are, sometimes, understood incompletely, as well as the basic properties, often, are investigated insufficiently, the main feature of these models is characterized by the factor that they all are of *non-numerical nature*[2]. Thus, searching methods form some principal research technology in resolving of a great number of fundamental and applied problems, both, while, algorithmic as well as inherent complexity analysis aspects of these problems are often kept into the backgrounds. Possibly, just these circumstances have outlined the frames of Automata Theory, in which (an abstract) automaton is investigated as some system with no anticipation inherent in it (see, for example, [1-5]). Of course, this point of view can be easily justified in terms of any axiomatic approach to Systems Theory[3]. Indeed, any automaton can be represented in the form of some discrete system, such that its response $\mathbf{y}(1)\mathbf{y}(2)\ldots\mathbf{y}(n)$ $(n \in \mathbf{Z}_+)$ to the input sequence $\mathbf{x}(1)\mathbf{x}(2)\ldots\mathbf{x}(n)$ is determined via some binary relation $\rho \subset \mathbf{X}^* \times \mathbf{Y}$, i.e. $(\mathbf{x}(1)\mathbf{x}(2)\ldots\mathbf{x}(i), \mathbf{y}(i)) \in \rho$ for all $i = 0,1,\ldots,n$. While ρ, $\mathbf{X}$ and $\mathbf{Y}$ are considered

[1] This research was supported, in part, through the Project 'Logic Approach in Dynamic Systems' Control', Reg.: 01024565

[2] I.e. they are presented via tables, graphs, logical equations and so on.

[3] It is worth to note that no unified axiomatic approach in Systems Theory is generally accepted, till now. Nevertheless, different systems of axioms (see, for example [6,7]) are consistent each with the others.

as abstract relation and sets, it is very difficult to develop any explicit form of incursive relations[4], connected with automata[5], similar to the ones successfully developed for systems of numerical nature in [8,9]. Besides, *the discontinuous*, inherent to finite automata makes it impossible to apply in Finite Automata Theory any manipulations, connected with the lim operation. The last factor complicates excessively any attempt to investigate anticipation for automata-based systems. Nevertheless, fundamental Problems, connected with automata-based systems' control lead directly to the notion of *an anticipation*[6] via decision-making process. Indeed, *closed-loop system*, proposed in [11] (see Fig. 1), *accumulates multiple choices creation with one choice's selection* and "**Thus such systems are implicit anticipatory systems because they evolve, from an initial state to a final state which is implicitly embedded in them. In an epistemic way, such implicit anticipatory systems evolve "as if they know their future"**" (see [8], p. 5). It is worth to note that, in realty, there exists very complicated knot of different types of anticipation, connected with automata-based systems. Indeed, if automata-based system is the fundamental one[7], then we deal with the strong anticipation in its pure form, as a rule. If, on contrary, an automata-based system is applied in the role of a model for some discrete device, then model-dependent anticipation can be the weak anticipation only with respect to the modelled device.

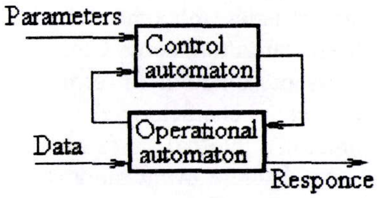

Fig. 1. Glushkov's closed-loop interaction.

The main aim of the paper is to investigate basic strong anticipation's characteristics, inherent to fundamental automata-based systems' control's Problems. The rest of the paper is organized as follows. In Section 2 the nature of oscillations for automata-based systems is discussed. In Section 3 the role of an incursivity in linear automaton's Controllability/Observability Problem's resolving is outlined. In Section 4 weakly initialized finite automaton's internal states' identification Problems' resolving is presented uniformly in terms of walks' strategies' design for some directed labeled multigraph with *shaded* vertices' labels. Anticipation is implemented into designed strategies via the walks' targets. Complexity of these strategies' design is investigated. In Section 5 resolving of supervisor's control's problems' resolving for any discrete event system (DES), presented via automata-based model[8] is outlined in terms of *forced* walks' strategies' design for some directed labeled multigraph. Completed tasks' maximal lan-

[4] I.e. relations of the form $w(t+1) = f(\ldots, w(t-2), w(t-1), w(t), w(t+1), w(t+2), \ldots; \mathbf{p})$, where the variable w denotes the state of the system and the variable $\mathbf{p}$ denotes the parameters (see [8], for example).

[5] Taking into account deep inherent links between finite automata and neural nets it looks very important, attractive and perspective an attempt to investigate incursivity backgrounds for McCulloch and Pits neurons, undertaken in [10].

[6] As well as to oscillations, which, sometimes, are directly connected with Chaos.

[7] I.e. this system is some specific form of an algorithm.

[8] I.e. the so-called RW-approach (see, for example, [12]).

guage's presentation via a closed-loop behavior results into anticipation and forms the designed supervisor's structure's skeleton. In Section 6 winning strategies' design for games on a graph is analyzed via design of multi-headed Turing Machine with an arbiter and independently controlled heads. This leads to some characteristics of distributed computing anticipation. Final remarks are given in the concluding section 7.

2 The Nature of Oscillations

Synchronous finite automaton is determined as a system $M = (Q, X, Y, \delta, \lambda)$, where Q, X and Y are finite sets, namely, correspondingly, the set of states, the input alphabet and the output alphabet, $\delta : Q \times X \to Q$ is the transition mapping and $\lambda : Q \times X \to Y$ is the output mapping. The simplest class of finite automata is formed by *autonomous automata*, i.e. the ones, such that $|X| = 1$[9]. In Fig. 2 some autonomous finite synchronous automaton, such that $Q = \{1, \ldots, 16\}$ and $Y = \{0,1,2\}$ is presented via directed labelled multigraph. Autonomous finite synchronous automata can illustrate

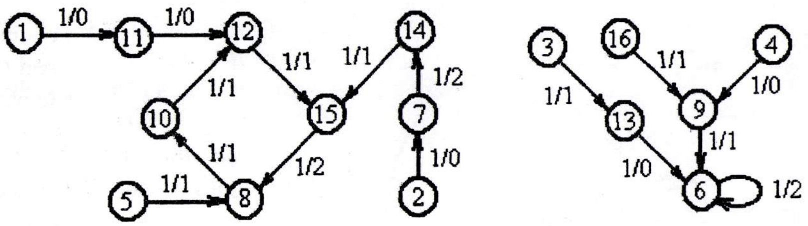

Fig. 2. An example of autonomous finite synchronous aotomaton's presentation via directed labelled multigraph (vertices' labels denote the automaton's states, while the arc's label *x/y* denotes an input-output pair) .

strong anticipation characteristics naturally. Indeed, the validity of the thesis "**The future states of a deterministic system are essentially dependent of both the initial and final conditions**" (see [8], p. 5) is evident via dealing with *super-strings*[10], since in this case the behavior of any autonomous finite synchronous automaton is completely determined by the equation

$$q(t+1) = f(q_{in}, Q_{fin}, t), \qquad (1)$$

where $q(t+1)$ is the state at instant $t+1$, q_{in} is the initial state and Q_{fin} is the *limit subset of states*[11]. Since for any autonomous automaton there exists the single input super-string 11...1..., it is not included into the equation (1).

[9] As a rule, it is supposed in this case, that $X = \{1\}$.

[10] I.e. infinite input sequences (see [3], for example).

[11] I.e. Q_{fin} consists all states that are reached infinitely many times, if input super-string is applied.

It is worth to note that for any initial and final conditions the equation (1) determines a sequence of states, which is either the empty one, or an infinite one. In the last case the generated super-string of states $q(1), q(2), \ldots, q(n), \ldots$ is *a periodic* one, i.e. there exist positive integers $i_0, l \leq |Q|$, such that for all $i, j \in \mathbf{N}$, if $i, j \geq i_0$ then $q(i) = q(j)$ if and only if $i \equiv j (\bmod l)$. For any autonomous finite synchronous automaton some periodic output super-string[12] corresponds to any periodic states' super-string. Thus, some specific oscillations form an inherent characteristic for any autonomous finite synchronous automaton's behavior. It is evident that these oscillations lead to *strong anticipation* for autonomous finite synchronous automata.

Possibly, the first attempts to present formally oscillations for any discrete device, were made in the sixties years of the XX-th Century and were connected with the extraction of *an operation mode*. These attempts have resulted into the notion of the *finite asynchronous Moore's type's automaton*.

Example. Some finite asynchronous Moore's type's automaton M_1 is presented in Fig. 3. The set of states of the automaton M_1 is partitioned into the two subsets. The first subset consists of *stable* states (these are the states 1, 4 and 5), while the second one consists of *unstable* states (these are the states 2, 3, 5, 7 and 8).

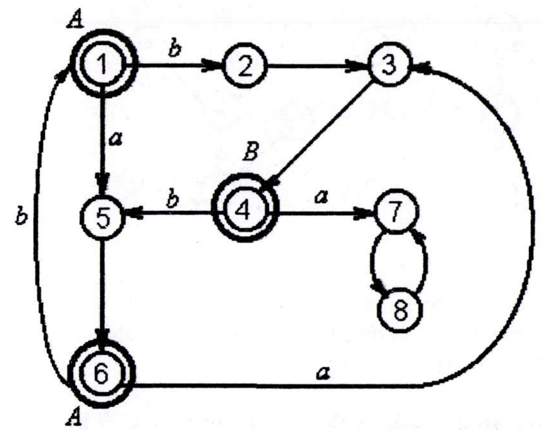

Fig. 3. Asynchronous automaton M_1.

Let the current stable state of the automaton M_1 be the state 1 and the input symbol a be applied. Then the automaton M_1, executing the transition through the unstable state 5, would reach the stable state 6 and response with the output symbol A. Thus, the described computing is an element of *the operation mode for the automaton* M_1.

Let the current stable state of the automaton M_1 be the state 4 and the input symbol a be applied. Then transitions via the cycle generated by the unstable states 7 and 8 would not terminate at all. Thus, the described computing is not an element of *the operation mode for the automaton* M_1.

Oscillations, like the ones described in Example, can take the place in realty for any asynchronous automaton's implementation via a discrete device. Thus, these oscil-

[12] I.e. the automaton's response.

lations lead to *strong anticipation* for asynchronous automata. Besides, these oscillations are the source of real Chaos, inherent to automata based systems.

Absolutely different type of oscillations is connected with discrete devices' *asynchronous logical simulation*. The basic idea of this process can be described as follows (see Fig. 4). It is designed some *Library*, consisting of basic elements' behavior's

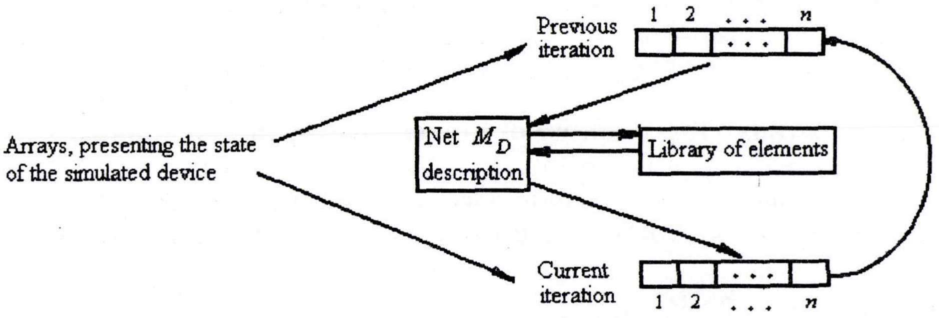

Fig. 4. Scheme of an asynchronous logical simulation of a discrete device.

descriptions. These descriptions are presented in the form of some instructions, written in Ternary Logic, as a rule[13]. Simulated device D is presented via some net M_D composed from basic elements. Let some external inputs' sequence $\mathbf{x}(1)\mathbf{x}(2)\ldots\mathbf{x}(n)$ be applied to the net M_D. To compute the response of the net M_D to any input $\mathbf{x}(i)$ $(i = 0,1,\ldots,n)$, some iterated process starts in the state the net M_D has reached after computing the response to the sequence $\mathbf{x}(1)\mathbf{x}(2)\ldots\mathbf{x}(i-1)$[14]. Two arrays are used to present the state of the net M_D via this process. The 1st array presents yet computed state, while the 2d-array presents the state that would be reached via signal's propagation through some selected element[15]. If the values of the arrays coincide, then the computing terminates. Otherwise, the 2d array is rewritten into the 1st one and the next iteration takes its place. It is evident, that some oscillations are possible[16]. These oscillations are the inherent ones for the applied model M_D only, as a rule, and there may be no oscillations at all in the simulated device D. Thus, *strong anticipation* of the simulation system can be, sometimes, resulted only into *weak anticipation* with respect to the modelled discrete device. It is evident that the described above fractal behavior of a model M_D is initialized by the attempt to present any time instant via some number of itera-

[13] The values 0 and 1 are used as usual Logical values, while the value u is used to present some *uncertainty*.

[14] It is supposed that the initial values of states of all elements of the net M_D are equal to u, as a rule.

[15] The order of elements' activation for the net M_D is fixed, as a rule.

[16] The sufficient condition is that at any iteration the values of the arrays differ from each other.

tions during the process of computing of current state of device D. Thus, fractal behavior can only indicate, sometimes, that the selected model is inadequate.

Remark. It is worth to note, that the similar situation often takes the place, when fractal behavior of numerical recursive models is investigated for applied problems[17] and no analysis for the values of parameters that lead to fractal behavior is given in terms of real investigated problem.

Thus, different types of oscillations are connected with automata-based systems. Some of them lead to weak anticipation with respect to a modelled system, while the others lead to strong anticipation, if automata-based system is a fundamental one.

3 Controllability/Observability of Linear Synchronous Automata

Let some finite field $\boldsymbol{F}=(F,+,\cdot)$ be fixed. Any finite *synchronous linear automaton* (over the field $\boldsymbol{F}$) M can be presented via some system of recurrent relations (see, for example, [13])

$$\begin{cases} \mathbf{s}(n+1)=\mathbf{A}\cdot\mathbf{s}(n)+\mathbf{B}\cdot\mathbf{x}(n) \\ \mathbf{y}(n)=\mathbf{C}\cdot\mathbf{s}(n)+\mathbf{D}\cdot\mathbf{x}(n) \end{cases} \quad (n\in\mathbf{N}),$$

where a state $\mathbf{s}$, an input $\mathbf{x}$ and an output $\mathbf{y}$ are elements of the corresponding vector-spaces over the field $\boldsymbol{F}$, i.e. $\mathbf{s}\in\boldsymbol{F}^k$, $\mathbf{x}\in\boldsymbol{F}^l$ and $\mathbf{y}\in\boldsymbol{F}^m$, and $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, $\mathbf{D}$ are, correspondingly, $(k\times k)$-matrix, $(k\times l)$-matrix, $(m\times k)$-matrix and $(m\times l)$-matrix, all over the field $\boldsymbol{F}$. Let $\mathbf{A}$ be some nonsingular matrix. Then we get

$$\mathbf{s}(n)=\mathbf{A}^{-1}\cdot\mathbf{s}(n+1)-\mathbf{A}^{-1}\cdot\mathbf{B}\cdot\mathbf{x}(n). \tag{2}$$

Sequentially substituting into (2) the values $n-1$, $n-2$, ..., 2, 1 instead of the value n, we get the following system of relations

$$\begin{cases} \mathbf{s}(n)=\mathbf{A}^{-1}\cdot\mathbf{s}(n+1)-\mathbf{A}^{-1}\cdot\mathbf{B}\cdot\mathbf{x}(n) \\ \mathbf{s}(n-1)=\mathbf{A}^{-1}\cdot\mathbf{s}(n)-\mathbf{A}^{-1}\cdot\mathbf{B}\cdot\mathbf{x}(n-1) \\ \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \\ \mathbf{s}(2)=\mathbf{A}^{-1}\cdot\mathbf{s}(3)-\mathbf{A}^{-1}\cdot\mathbf{B}\cdot\mathbf{x}(2) \\ \mathbf{s}(1)=\mathbf{A}^{-1}\cdot\mathbf{s}(2)-\mathbf{A}^{-1}\cdot\mathbf{B}\cdot\mathbf{x}(1) \end{cases}$$

Let the 1st relation be multiplied by $(\mathbf{A}^{-1})^{n-1}$, the 2-d relation be multiplied by $(\mathbf{A}^{-1})^{n-2}$, ..., the $(n-1)$-th relation be multiplied by $\mathbf{A}^{-1}$. Computing the sum of all these relations, we get the following incursive relation

$$\mathbf{s}(1)=(\mathbf{A}^{-1})^n\cdot\mathbf{s}(n+1)-\sum_{i=1}^{n}(\mathbf{A}^{-1})^i\cdot\mathbf{B}\cdot\mathbf{x}(i). \tag{3}$$

Let M be some *weakly initialized automaton*[18]. We set

$$\mathbf{U}_i=\{\mathbf{A}\cdot\mathbf{s}+\mathbf{B}\cdot\mathbf{x}(i)\,|\,\mathbf{C}\cdot\mathbf{s}+\mathbf{D}\cdot\mathbf{x}(i)=\mathbf{y}(i)\;\&\;\mathbf{s}\in\mathbf{U}_{i-1}\}\ (i=1,\dots,n).$$

[17] For example, in Economics area.

[18] I.e. it is fixed some initial condition of the form $\mathbf{s}(1)\in\mathbf{U}_0$ $(\mathbf{U}_0\subseteq\boldsymbol{F}^k, |\mathbf{U}_0|\geq 2)$.

Thus, the final state of the automaton M would satisfy to the condition $\mathbf{s}(n+1) \in \mathbf{U}_n$, i.e. *the exactness* of resolving of Controllability Problem for the automaton M is characterized by the set $\mathbf{U}_n$. In particular, the Controllability Problem for the automaton M would be resolved uniquely, if $|\mathbf{U}_n| = 1$. Indeed, to convert any state $\mathbf{s}(n+1)$ into any prescribed state $\mathbf{s}$, it is sufficient to apply to the automaton M any input $\mathbf{x}$, such that $\mathbf{B} \cdot \mathbf{x} = \mathbf{s} - \mathbf{A} \cdot \mathbf{s}(n+1)$.
Relation (3) implies that *the exactness* of resolving of Observability Problem for the automaton M is characterized by the set
$\{\mathbf{s} \mid \mathbf{s} = (\mathbf{A}^{-1})^n \cdot \mathbf{s}_1 - \sum_{i=1}^{n} (\mathbf{A}^{-1})^i \cdot \mathbf{B} \cdot \mathbf{x}(i) \,\&\, \mathbf{s}_1 \in \mathbf{U}_n\}$.
In particular, the Observability Problem for the automaton M would be resolved uniquely, if $|\mathbf{U}_n| = 1$.

Thus, incursive relations form the strong base for developing of sufficiently powerful tools for Controllability/Observability analysis of discrete linear systems.

4 Identification of Finite Synchronous Automata's States

For any finite synchronous automaton $M = (Q, X, Y, \delta, \lambda)$[19] we set
$(X \times Y)(q, q') = \{(x, y) \in X \times Y \mid (\delta(q, x) = q') \,\&\, (\lambda(q, x) = y)\}$,
$wght(q, q') = |\{x \in X \mid \delta(q, x) = q'\}| \quad (q, q' \in Q)$.
It is well known that any automaton M can be presented via some directed multigraph G_M with arcs labeled by elements of the set $X \times Y$, such that:

1) the set of vertices of G_M is Q;

2) the set of arcs of G_M consists of $|Q| \cdot |X|$ elements, determined in the following way: the number of copies of an arc (q, q') $(q, q' \in Q)$ equals to $wght(q, q')$[20] and different copies of an arc (q, q') are labeled by different elements of the set $(X \times Y)(q, q')$.

Any input sequence $p = x_1 x_2 \ldots x_l \in X^+$ can be interpreted as some *strategy of a walk* of the length l in G_M. Indeed, let q_Λ[21] be any vertex in G_M. There exists the single arc $\overrightarrow{e_1}$ started in q_Λ and labeled by the element $(x, y) \in X \times Y$, such that $x = x_1$. Let the arc $\overrightarrow{e_1}$ be ended in a vertex q_{x_1}. There exists the single arc $\overrightarrow{e_2}$ started in q_{x_1} and labeled by the element $(x, y) \in X \times Y$, such that $x = x_2$. Let the arc $\overrightarrow{e_2}$ be ended in a vertex $q_{x_1 x_2}$, and so on. Thus, for any vertex q_Λ it is uniquely determined in G_M the walk

---

[19] It was pointed in Chapter 2 that Q is the set of states, X is the input alphabet, Y is the output alphabet, $\delta: Q \times X \to Q$ is the transition mapping, $\lambda: Q \times X \to Y$ is the output mapping.

[20] The identity $wght(q, q') = 0$ implies that there is no arc (q, q') in G_M, at all.

[21] Λ denotes *the empty string*.

$\pi_{q_\Lambda} = q_\Lambda, \overrightarrow{e_1}, q_{x_1}, \overrightarrow{e_2}, q_{x_1x_2}, \ldots, \overrightarrow{e_{l-1}}, q_{x_1\ldots x_{l-1}}, \overrightarrow{e_l}, q_{x_1\ldots x_l}$

of length l, started in q_Λ. *The characteristic* of this walk is determined to be the output sequence $y_1y_2\ldots y_l \in Y^*$, such that $(x_i, y_i) \in X \times Y$ is the label of the arc $\overrightarrow{e_i}$ for all $i = 1,\ldots,l$.

Let (M,Q_0) [22] be some *weakly initialized automaton* (w.i.a.) and let $\widetilde{\delta}$ and $\widetilde{\lambda}$ be the extensions of the mappings δ and λ to the set $Q \times X^*$, determined in the usual way. An input sequence $p \in X^+$ is called to be:

1) *a distinguishing* one, if $(\forall q', q'' \in Q_0)(\widetilde{\lambda}(q',p) = \widetilde{\lambda}(q'',p) \Rightarrow q' = q'')$;

2) *a homing* one, if $(\forall q', q'' \in Q_0)(\widetilde{\lambda}(q',p) = \widetilde{\lambda}(q'',p) \Rightarrow \widetilde{\delta}(q',p) = \widetilde{\delta}(q'',p)$;

3) *a synchronizing* one, if $(\forall q', q'' \in Q_0)(\widetilde{\delta}(q',x) = \widetilde{\delta}(q'',x))$.

The Problems of design of these identifying sequences for a w.i.a. (M,Q_0) can be characterized in terms of the following walks' strategies' design for G_M, under the supposition that the vertices' labels in G_M are *blotted*[23]:

1) design of any distinguishing sequence is reduced to design of some strategy $p \in X^+$ of the walk in G_M, such that in the result of the walk π_q in G_M, started in any vertex $q \in Q_0$ and carried out in accordance with the strategy p, the vertice q would be identified uniquely in the result of the analysis of the characteristic of the walk π_q, only;

2) design of any homing sequence is reduced to design of some strategy $p \in X^+$ of a walk in G_M, such that in the result of the walk π_q in G_M, started in any vertex $q \in Q_0$ and carried out in accordance with the strategy p, the endvertex of the walk π_q would be identified uniquely in the result of the analysis of the characteristic of the walk π_q, only;

3) design of any synchronizing sequence is reduced to design of some strategy $p \in X^+$ of a walk in G_M, such that in the result of a walk π_q in G_M, started in any vertex $q \in Q_0$ and carried out in accordance with the strategy p, the endvertex of walk π_q would be the same.

Any of the above determined strategy of a walk in G_M would be called to be:

1) *an optimal* one, if p is some shortest identifying sequence for (M,Q_0);

2) *an irreducible* one, if any sequence obtained in the result of deleting in p some letters would not an identifying sequence for the w.i.a. (M,Q_0).

---

[22] Q_0 $(Q_0 \subseteq Q, |Q_0| \geq 2)$ is *the set of initial states*.

[23] In another words, vertices' labels are unobservable.

It is worth note high complexity of determined walks' strategies' design. Indeed, the following estimations of *Shannon's functions* for shortest identifying sequences' lengths has been established[24]:

$\mathsf{L}^{h}_{k.m.n}(r) = 0.5\cdot(2\cdot k - r)\cdot(r-1)$ $(r\in\{2,\ldots,k\})$ (*T.N. Hibbard*, in 1965),

$$\mathsf{L}^{d}_{k.m.n}(r) \geq \begin{cases} \binom{k-1}{r-1} \ , & if \ r\in\{2,\ldots,\lfloor 0.5\cdot k\rfloor\} \\ \binom{k-2}{\lfloor 0.5\cdot(k-2)\rfloor}, & if \ r\in\{\lfloor 0.5\cdot k\rfloor+1,\ldots,k-1\} \\ 3^{\left\lfloor \frac{1}{6}\cdot k\right\rfloor} \ , & if \ r=k \end{cases}$$ (*M.N. Sokolovsky*, 1976),

$\log_3 \mathsf{L}^{d}_{k.m.n}(k) \sim \dfrac{k}{6}$ $(k\to\infty)$ (*I.K. Rystsov*, 1978),

$\mathsf{L}^{d}_{k.2.n} \geq e^{O(\sqrt{k})}$ $(k\to\infty)$ (*V.G. Skobelev*, 1987),

$\mathsf{L}^{s}_{k.2.n} \geq e^{O(\sqrt{k})}$ $(k\to\infty)$ (*V.G. Skobelev*, 1987).

Identifying sequences are intended for *preset experiments* with the w.i.a. (M,Q_0). In accordance with this factor, we refer to the described above strategies of walks in G_M as to *preset* strategies. Another type of experiment with the w.i.a. is *an adaptive* one. Automata-experimenters' design methods for this type of experiments were developed in [14][25]. It is evident that we can refer to any automaton-experimenter as to *an adaptive strategy* of some walk in G_M. Moreover, this type of walks can be naturally reduced to some problems, connected with *the behavior of an automaton in a maze*.

In the above described experiments some single identifying sequence or, correspondingly, a single automaton-experimenter were used. This type of experiments with the w.i.a. is referred to be *simple* ones. In accordance with this factor, we refer to the described above strategies of walks in G_M as to *simple* ones. If in an experiment with the w.i.a. it is used some set consisting of at least of two sequences (correspondingly, at least of two automata-experimenters), each intended to obtain some partial solution, and all these partial solutions result into some complete solution, then the experiment with the w.i.a. is called to be *a multiple* one. We refer to corresponding strategies of walks in G_M as to *cooperative* ones. It is evident that this type of walks can be naturally reduced to some problems, connected with the behavior of *a group of* (possibly, *interacting*) *automata in a maze*.

Thus, strong anticipation forms inherent characteristics for sufficiently wide class of Problems of Discrete Mathematics, Graph Theory and Automata Theory, connected with decision-making.

[24] It is supposed that $|Q|=k$, $|X|=m$, $|Y|=n$, $|Q_0|=r$.

[25] General approach for automata-experimenters' design for resolving discrete problems is systematically developed in [15].

5 Discrete Event Systems (DES)

Developed in [12] approach for DES' analysis is based on presenting of a DES via an acceptor $M = (Q, \Sigma, \delta, q_0, Q_m)$, where Q and Q_m $(Q_m \subseteq Q)$ are, correspondingly, the set of states and the set of marker states, q_0 $(q_0 \in Q)$ is the initial state, Σ is an alphabet of event labels, and $\delta : Q \times \Sigma \to Q$ is, possibly, partial transition function. An acceptor M is characterized by two subsets of Σ^*, namely, the closed behavior
$\mathbf{L}(M) = \{s \in \Sigma^* \mid \tilde{\delta}(q_0, s) \text{ is determined}\}$
and the language of completed tasks
$\mathbf{L}_m(M) = \{s \in \Sigma^* \mid \tilde{\delta}(q_0, s) \in Q_m\}$.
The alphabet of event labels Σ is partitioned into nonempty sets of controllable events Σ_c and the set of uncontrollable events Σ_{uc}. The first ones can be either enabled or disabled by some external agent, while the second ones cannot be prevented from occurring and therefore are considered to be permanently enabled. The feedback control of the event-disablement mechanism is based on the notion of a supervisor $\mathbf{S} = (A, \varphi)$ upon M, where $A = (U, \Sigma, \psi, u_0, U_{fin})$ is an acceptor and a mapping $\varphi : U \times \Sigma \to \{0,1\}$ satisfies to the condition: $\varphi(u, \sigma) = 1$, if $\sigma \in \Sigma_{uc}$ and $\varphi(u, \sigma) \in \{0,1\}$, if $\sigma \in \Sigma_c$ (i.e. 1 is interpreted as 'enable', while 0 is interpreted as 'disable'). Thus the behavior of a closed-loop system is represented via some automaton $\langle M \mid \mathbf{S} \rangle$ (see, Fig. 5).

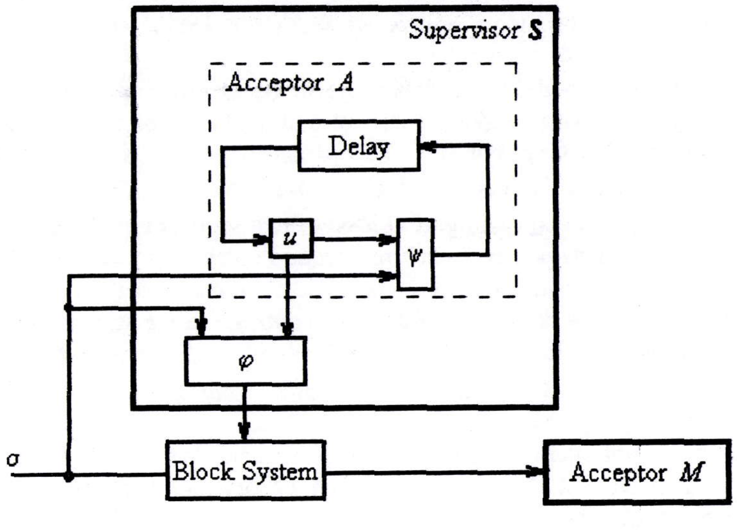

Fig. 5. Closed-loop system $\{M \mid \mathbf{S}\}$.

Slight modification of the notion of a DES, proposed in [16], leads to some simplifications of DES' analysis, and makes it possible to operate with different types of concurrency, as well as to apply efficiently Algebra of Acceptors, developed completely in [3][26]. It is proposed in [16] to deal with an acceptor $M = (Q, \Sigma, \delta, q_{in}, Q_m, q_{fin})$, where Q and Q_m $(Q_m \subseteq Q)$ are, correspondingly, the set of states and the set of marker states, Σ is an alphabet of event labels, q_{in} and q_{fin} $(q_{in}, q_{fin} \in Q)$ are, correspondingly, the initial and the final states and $\delta \subseteq Q \times (\Sigma \cup \{\Lambda\}) \times Q$ is the transition relation, such that $(q, \Lambda, q_{fin}) \in \delta \Leftrightarrow q \in Q_m$ and $(\forall \sigma \in \Sigma \cup \{\Lambda\})(\forall q \in Q)(q_{fin}, \sigma, q \notin \delta)$.

Remark. It is worth to note that design of any extremal (in particular, the optimal) supervisor, leads to high complexity of a system $\langle M | \mathbf{S} \rangle$. Indeed, the number of supervisor's states is estimated by 2^k (where k is the number of states of the acceptor M), and it is well known that this estimation can be reached.

For any DES $M = (Q, \Sigma, \delta, q_{in}, Q_m, q_{fin})$ we set
$(\Sigma \cup \{\Lambda\})(q, q') = \{\sigma \in (\Sigma \cup \{\Lambda\} \,|\, ((q, \sigma, q') \in \delta\}$,
$wght(q, q') = |\, \{\sigma \in \Sigma \cup \{\Lambda\} \,|\, (q, \sigma, q') \in \delta\}$ $(q, q' \in Q)$.
Any DES M can be presented via directed multigraph G_M with arcs labeled by elements of the set $X \times Y$, such that:

1) the set of vertices of G_M is Q;

2) the set of arcs of G_M consists of $|\delta|$ elements, determined in the following way: the number of the copies of an arc (q, q') $(q, q' \in Q)$ equals to $wght(q, q')$[27] and different copies of an arc (q, q') are labeled by different elements of the set $(\Sigma \cup \{\Lambda\})(q, q')$.

The Problem of design of a supervisor $\mathbf{S} = (A, \varphi)$ for a DES M can be reduced to some *forced* walk's adaptive strategy's design for G_M, under the supposition that the vertices' labels in G_M are *blotted*. Indeed[28], let a supervisor $\mathbf{S}$ be supplied with some events' sequence $\sigma_1 \sigma_2 \ldots \sigma_n \in \Sigma^+$, while it is visiting the vertice q_{in}. Then supervisor's actions are determined in accordance with the following rules:

1. If $\mathbf{S}$ is visiting the final vertex q_{fin}, then computing terminates and the walk halts.

2. Let $\mathbf{S}$ have terminated successfully computing, connected with the initial fragment $\sigma_1 \sigma_2 \ldots \sigma_{i-1}$ $(i = 1, \ldots, n)$ and $\mathbf{S}$ is visiting some vertex q $(q \neq q_{fin})$ of G_M. If $\sigma_i \in \Sigma$ is any event, such that $\sigma_1 \ldots \sigma_{i-1} \sigma_i \notin \mathbf{L}(\boldsymbol{M}_\pi)$ then all computing as well as a walk halts. Let $\sigma_i \in \Sigma$ be any event, such that $\sigma_1 \ldots \sigma_{i-1} \sigma_i \in \mathbf{L}(\boldsymbol{M}_\pi)$. If $\varphi(u, \sigma_i) = 1$, where u is the state of the acceptor A, then $\mathbf{S}$ is walking along some arc of G_M, started in q and labeled with the event σ_i (and possibly, along some sequence of sub-

[26] Developed in [3] algebra, being some variant of Kleene's algebra, is intended for algorithms' design.

[27] The identity $wght(q, q') = 0$ implies that there is no arc (q, q') in G_M, at all.

[28] For simplicity, it is supposed that there is no concurrency of any type in DES.

sequent arcs labeled by the empty sequence Λ). If $\sigma_i \in \Sigma_c$ and $\varphi(u,\sigma)=0$, then $\boldsymbol{S}$ terminates computing, connected with the initial fragment $\sigma_1\sigma_2\ldots\sigma_{i-1}\sigma_i$ and starts the processing of the next event.

Remark. It is evident that the above described adaptive strategy of a forced walk in a directed multigraph G_M is interpreted naturally as *a Game with The Nature*.

Thus, strong anticipation forms inherent characteristics for sufficiently wide class of Problems of control design for DES.

6 Games on a Graph

Any *Two-Players Game* on a graph can be presented via some system $\boldsymbol{g}=(P,F,G,p_{in},P_1^{win},P_2^{win})$, where P is a finite set of *positions*, $p_{in}\in P$ is *the initial position*, P_1^{win} and P_2^{win} $(P_1^{win},P_2^{win}\neq\varnothing; P_1^{win}\cap P_2^{win}=\varnothing; p_{in}\notin P_1^{win}\cup P_2^{win})$ are, the sets of *winning positions*, correspondingly, for the 1st and for the 2d Player, F and G are some sets of (possibly, partial) mappings of the set P into itself, called *the sets of moves*, correspondingly of the 1st and of the 2d Player. *A play* in a game $\boldsymbol{g}$ is determined to be any sequence

$$p_0^{(1)},f_1,p_0^{(2)},g_1,p_1^{(1)},f_2,p_1^{(2)},g_2,\ldots, \tag{4}$$

such that:

1) $p_0^{(1)}=p_{in}$ and $p_i^{(1)}\in Dom\,f_{i+1}$, $p_i^{(2)}\in Dom\,g_{i+1}$ for all $i=0,1,\ldots$;

2) positions $p_0^{(1)},p_1^{(1)},\ldots$, as well as positions $p_0^{(2)},p_1^{(2)},\ldots$ are pairwise different;

3) if current position is $p\in P_1^{win}\cup P_2^{win}$, then a play terminates.

Thus, all plays of a game $\boldsymbol{g}$ are partitioned into the following four sets

$$\Pi_1^{win}=\{p_0^{(1)},f_1,\ldots,p_{k-1}^{(1)},f_n,p_{k-1}^{(2)}\ (k\in\mathbf{N})\mid p_{k-1}^{(2)}\in P_1^{win}\},$$

$$\Pi_2^{win}=\{p_0^{(1)},f_1,\ldots,p_{k-1}^{(2)},g_k,p_k^{(1)}\ (k\in\mathbf{N})\mid p_k^{(1)}\in P_2^{win}\},$$

$$\widetilde{\Pi}_1=\{p_0^{(1)},f_1,\ldots,p_{k-1}^{(2)}\ (k\in\mathbf{N})\mid (p_{k-1}^{(2)}\notin P_1^{win})\,\&\,(\forall g\in G)(p_{k-1}^{(2)}\notin Dom\,g)\},$$

$$\widetilde{\Pi}_2=\{p_0^{(1)},f_1,\ldots,p_k^{(1)}\ (k\in\mathbf{N})\mid (p_k^{(1)}\notin P_2^{win})\,\&\,(\forall f\in F)(p_k^{(1)}\notin Dom\,f)\}.$$

To simplify reasoning, we restrict ourselves by dealing only with the games, such that $\widetilde{\Pi}_1=\widetilde{\Pi}_2=\varnothing$.

Any strategy of the 1st Player in a game $\boldsymbol{g}$ is determined to be some algorithm A_1, such that for any play (4) the identity

$$f_{i+1}=\mathsf{A}_1(p_i^{(1)})$$

holds for all $i=0,1,\ldots$. Similarly, any strategy of the 2d Player in a game $\boldsymbol{g}$ is determined to be some algorithm A_2, such that for any play (4) the identity

$$g_{i+1}=\mathsf{A}_2(p_i^{(2)})$$

holds for all $i=0,1,\ldots$.

Any play in a game $\boldsymbol{g}$ can be simulated naturally via computing carried out by some 2-dimensional *Turing Machine* (TM) with two *heads*, controlled by two different *control units* (see Fig. 6). Control units CU1 and CU2 (and the heads, controlled by

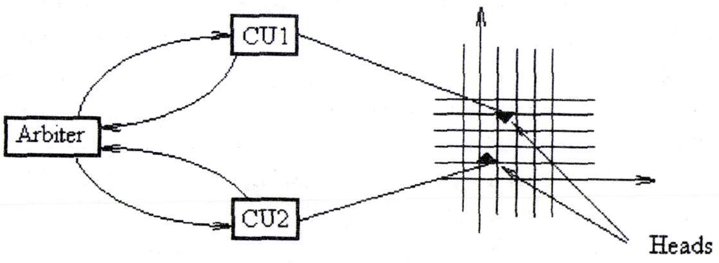

Fig. 6. Simulation of a play in a game $\boldsymbol{g}$ by 2-dimensional Turing Machine M, consisting of 2 heads, controlled by different control units.

these units) simulate, correspondingly the actions of the 1st and of the 2d Player. Moreover, control unit CU1 operates in accordance with some strategy A_1 of the 1st Player, while control unit CU2 operates in accordance with some strategy A_2 of the 2d Player. The initial configuration of TM M presents the initial position $p_{in} = p_0^{(1)}$ of the game $\boldsymbol{g}$. At initial instant of time *the Arbiter* initializes control unit CU1. The last carries out computing, that simulates the move $f_1 = A_1(p_{in})$. As soon as computing carried out by control unit CU1 is executed, CU1 informs the Arbiter and halts in the configuration, that presents the position $p_0^{(2)}$. If the play is not finished, then the Arbiter initializes control unit CU2. The last carries out computing, that simulates the move $g_1 = A_2(p_0^{(2)})$. As soon as computing carried out by control unit CU2 is executed, CU2 informs the Arbiter and halts in the configuration, that presents the position $p_1^{(1)}$. If the play is not finished, then the Arbiter initializes control unit CU1 and so on.

Remark. 1. The value 2 for the dimension of the tape of TM is selected only to use efficiently *visual aids*, viz. to stress that control units can use for computing any size of external memory, without any disturbance to each other. For example, without loss of generality it can be supposed that all positions of a play in a game $\boldsymbol{g}$ are presented in the 1st quadrant, while control units CU1 and CU2 use for computing cells disposed, correspondingly, in the 2d and in the 4th quadrants. It is also evident that control units CU1 and CU2 and the Arbiter can be joined into the single control unit. But in the result of this joining all clearness would be lost completely and some features, essential for the Players' strategies' design would be shaded.

2. It is evident, that any multi-headed TM, which heads are controlled independently by different control units, simulates distributed, in particular, parallel computing. The described above simulation extracts strong anticipation for sufficiently wide class of distributed computing.

For a game $\boldsymbol{g}$ some strategy A_1 of the 1st Player is called to be *a winning* one[29], if its applying guarantee that the play would be an element of the set Π_1^{win} for any strategy A_2 of the 2d Player. Similarly, some strategy A_2 of the 2d Player is called to be *a winning* one[30], if its applying guarantee that the play would be an element of the set Π_2^{win} for any strategy A_1 of the 1st Player. It is evident that design of winning strategy for the 1st Player can be reduced to design of control unit CU1, such that for any permissible control unit CU2 TM M would simulate some play, being the element of the set Π_1^{win}. Similarly, design of winning strategy for the 2d Player can be reduced to design of control unit CU2, such that for any permissible control unit CU1 TM M would simulate some play, being the element of the set Π_2^{win}.

It is worth to note, that some specific features are connected with games on a graph (see, for example, [16]).

Firstly, winning strategy's design for any game on a graph is connected with some walks' strategies' design, intended to deal with some *presentation of a graph*[31].

Secondly, for a game on a graph *the tree of a game* often cannot be reduced into an automaton with the number of states, comparable with the size of analyzed game. This implies that complexity of control units CU1 and CU2 of TM M is sufficiently high, as a rule.

Thirdly, sufficiently wide class of applied Problems can be reduced to some game on a graph, such that the aim of the 1st Player is to design this or the other *graph-theoretic structure*[32], while the actions of the 2d Player can prevent from the 1st Player's efforts. This class of games is intended to model any situation, when the 1st Player represents implementation of some means to provide these or the others conditions for investigated object or process, while the 2d Player represents some instability actions of the environment, i.e. design of winning strategy forms some base for *decision-making in instability environment*.

Thus, strong anticipation forms inherent characteristics for games on graphs, as well as for some types of distributed computing.

7 Conclusions

In the given paper it was made an attempt to establish that strong anticipation is an inherent characteristic for some fundamental Problems, connected with automata-based systems' control. These systems are discrete ones of non-numerical nature. Thus, the Problem of working-out of some approach for efficient design of numerical incursive relations for automata-based systems is actual. Some backgrounds for this Problem's resolving, developed in [10], and investigation of links between neural nets and finite automata can form strong base for systematic oscillations' nature's analysis, in-

[29] With respect to the 1st Player.
[30] With respect to the 2d Player.
[31] In contradistinction to walks' strategies' design for a graph, presented in the previous two Chapters.
[32] I.e. some path, cycle, tree and so on.

herent to automata-based systems, as well as for extraction and classification of different types of strong and weak anticipation.

It seems that there must be investigated in details anticipation characteristics of automata-based systems, presented via relations, designed in terms of vector spaces over finite fields. These investigations could be very useful for establishing of some deep links between ordinary numerical incursive relations and incursive relations, designed for systems, presented via tables or graphs.

Automata-based systems' control's design determines an important class of Problems with strong anticipation in the role of basic inherent characteristic. It is evident that sometimes, like in resolving Problems of states' identification for finite automaton, strong anticipation can be easily characterized either by initial conditions, or by final conditions, or by initial and finite conditions, both. Unfortunately, anticipation connected with supervisor's design can't be described so easily and its backgrounds must be investigated deeply.

Established links between design of control for automata-based systems and design of walks' strategy on a graph is the evidence of the factor that strong anticipation is an inherent characteristic for a large number of fundamental Discrete Mathematics' Problems. This statement is also justified by proposed technique for simulation of any game on a graph via multi-headed TM, with heads controlled independently by different control units. Besides, the Problem of investigation of the role of strong anticipation in distributed computing seems to be very important.

Above pointed trends for strong anticipation investigation form some base for future research.

Acknowledgements

The Author is grateful to Dr. Daniel M. Dubois (Liege, Belgium), the Person who has attracted the Author's attention to a knot of amazing Problems, connected with the notion of anticipation and whose efforts and patience were the main source for completing of the given paper.

References.

1. *Hennie F.C.* Finite-State Models for Logical Machines. John Wiley&Sons INC.: NY, 1962, 466p.
2. *Kohavi Z.* Switching and Finite Automata Theory. McGrow-Hill Book Company: NY, 1970, 592p.
3. *Trachtenbrot B.A., Barzdin Ya. M.* Finite automata (Analysis and Synthesis). Published by Nauka Company (Moscow, Russia), 1970, 400p. (in Russian).
4. *Eilenberg S.* Automata, Languages and Machines. Vol. **A**. Academic Press: NY, 1976, 451p.
5. *Eilenberg S.* Automata, Languages and Machines. Vol. **B**. Academic Press: NY, 1976, 387p.

6. *Mesarovich M.D., Takahara Y.* General Systems Theory. Mathematical Foundations. Academic Press, N.Y. Translated into Russian and Published by Mir Company (Moscow, Russia), 1978, 311p.
7. *Matrosov V.M., Anapolsky L.Yu., Vasil'ev S.N.* The Comparison Method in Mathematical Systems Theory. Published by Nauka Company (Novosibirsk, Russia), 1980, 480p.
8. *Dubois D.M.* Review of Incursive, Hyperincursive and Anticipatory Systems. – Foundation of Anticipation in Electromagnetizm // *Computing Anticipatory Systems: CASYS'99 – Third International Conference*. Edited by D.M. Dubois. Published by the American Institute of Physics, AIP Conference Proceedings 517, 2000, pp. 3-30.
9. *Dubois D.M.* Theory of Incursive Synchronization and Application to the Anticipation of Chaotic Epidemic // *International Journal of Computing Anticipatory Systems* (Published by asbl CHAOS, Belgium, ISSN 1373-5411, ISBN 2-9600262-3-3), vol. **10**, 2001, pp.3-18.
10. *Dubois D.M.* Hyperincursive McCulloch and Pitts Neurons for Designing a Computing Flip-Flop Memory // *Computing Anticipatory Systems: CASYS'99 – Second International Conference*. Edited by D.M. Dubois. Published by the American Institute of Physics, AIP Conference Proceedings 465, 2000, pp. 3-21.
11. *Glushkov V.M.* Digital Automata Synthesis. Published by FizMatGiz Company (Moscow, Russia), 1962, 400p. (in Russian).
12. *Ramage P.J., Wonham W.M.* Supervisory Control of a Class of Discrete Event Processes // *SIAM Journal of Control and Optimization*, vol. **25**, 1987, pp.206-230.
13. *Gill A.* Linear Sequential Machines. McGrow Hill Book Company, NY. Translated into Russian and Published by Nauka Company (Moscow, Russia), 1974, 287p.
14. *Skobelev V.G.* Design of Automata-Experimenters. *Methods of Diagnosis and Control for Complex Discrete Systems and Automata.* Published by Institute of Cybernetics of National Academy of Sciences of Ukraine (Kiev, Ukraine), 1973, pp. 32-44 (in Russian).
15. *Skobelev V.G.* Discrete Systems Analysis. Published by Institute of Applied Mathematics and Mechanics of National Academy of Sciences of Ukraine (Donetsk, Ukraine), 2002, 172p. (in Russian).
16. *Skobelev V.G.* Local Algorithms for Graphs. Published by Institute of Applied Mathematics and Mechanics of National Academy of Sciences of Ukraine (Donetsk, Ukraine), 2003, 217p. (in Russian).