Math. Maced.
Vol. 8 (2010)
79-94

PROFESSOR CUPONA AND THE FIRST LECTURE ON ABSTRACT AUTOMATA IN MACEDONIA

STEVO BOZINOVSKI

Dedicated to Academician Ǵorǵi Čupona

Abstract. This paper was presented as an invited plenary paper at the conference dedicated to Professor Cupona in September 2010. The paper honors the role of professor Gorgi Cupona in the early development of computer science and cybernetics in Macedonia. The period covered in this paper starts in 1968 when Dr Cupona organized the first seminar related to Cybernetics as well as the first lecture on abstract automata in Macedonia. The paper also describes an instance of collaboration of Dr Cupona with the students, from high school level to graduate level.

1. Introduction

[0]

This paper is dedicated to the memory of Professor Gorgi Cupona. The work addresses the influence and contribution of Professor Cupona to Computer Science and Cybernetics in Macedonia. It is a view of a high school student influenced by professor Cupona's vision, ideas, and acts.

The time period of collaboration between Dr Cupona and the author covered in this work is between 1968 (presenting first lecture on abstract automata in Macedonia) and 1982 (defending the PhD thesis). It is a period in which the author as a student (from high school to graduate level) was in contact with professor Cupona discussing various topics related to mathematics.

In the sequel we first observe the role of Dr Cupona in mathematics competitions and nurturing the high school talent. Then we give information about a scientific event, appearance of a book on cybernetics by Glushkov, which influenced organization of a seminar on cybernetics and also the first lecture on abstract automata in Macedonia. Then we give overview on abstract automata and we describe systems such as agents, abstract computational machines, and neural networks. Then we point out the engagement of Dr. Cupona in building a scientific infrastructure, the Mathematics Institute with Numeric Center. In the last chapter we describe

2000 *Mathematics Subject Classification.* Primary : algebra; Secondary: automata theory.
Key words and phrases. difference equations, automata, Turing machines, pattern recognition, integer programming, self-evaluating automata, emotional automata.

[0]Part of the research presented in this paper was funded by the Fulbright program, 1980-1981.

a result of the motivation and encouragement provided by Dr Cupona to the contribution of the author in the mentioned time period to self-organizing systems in terms of abstract automata.

2. 1960's: High School Competitions in Mathematics in Macedonia

Many high school students met Dr Cupona for the first time during preparations for mathematics competitions. Here we will mention a generation of students including Smile Markovski, Dimitar Altiparmakov, Tome Mickovski, Risto Ciconkov, Biljana Arsova, Eli Delidzakova, Gjorgi Josifovski, Stevo Bozinovski, among many others. Here we will also mention the "idols" Smilka Zdravkovska and Viktor Urumov, who achieved to participate at the world mathematics competition in Moskow. Three of the mentioned same generation students, Dimitar, Smile and Stevo, participated at the federal competitions in Belgrade. The system of organized work with high school students included high school mathematics professors who worked with students preparing them for the competitions. One such example was Gorica Ilieva, the wife of Professor Cupona. She had dear personality and knowledge to attract students to engage in mathematics. The author had a privilege and a pleasure to be one of those students.

Part of preparation for mathematics competitions were various mathematics schools organized by Dr. Cupona. During those activities the interested high school students were exposed to mathematics lectures beyond standard high school curriculum. For example, at that time the concept of sets was not in regular high school mathematics curriculum. It was taught in those extracurricular activities, along with determinants, matrices, etc. The instructors at the mentioned mathematics schools were university professors, for example Naum Celakovski, Zivko Madevski, Aleksandar Samardziski, Branko Trpenovski, among others.

3. 1966: Event in Science: The Book on Cybernetics by Glushkov

An important event in science turned out to be influential in the development of mathematics, computer science, and cybernetics in Macedonia. In 1966 the book *Introduction to Cybernetics* by Glushkov appeared offering a rather radical new view toward Cybernetics. While the previous view was related to control theory, the new book introduced Cybernetics through theory of algorithms, languages, and automata. The book was translated in several languages, for example in USA in the same year [16]. The book appeared at the book fair in Skopje in 1967, translated by Rajko Tomovic and Momcilo Uscumlic. Several Fair visitors bought the book among them Dr. Cupona and, independently, the author of this paper. Since the book is important for this paper we give here the chapter overview:

Ch1. The abstract theory of algorithms
Ch2. Boolean functions and propositional calculus
Ch3. Automata theory
Ch4. Self-organizing systems
Ch5. Electronic digital computers and programming

Ch6. The predicate calculus and the automation of the processes of scientific discovery

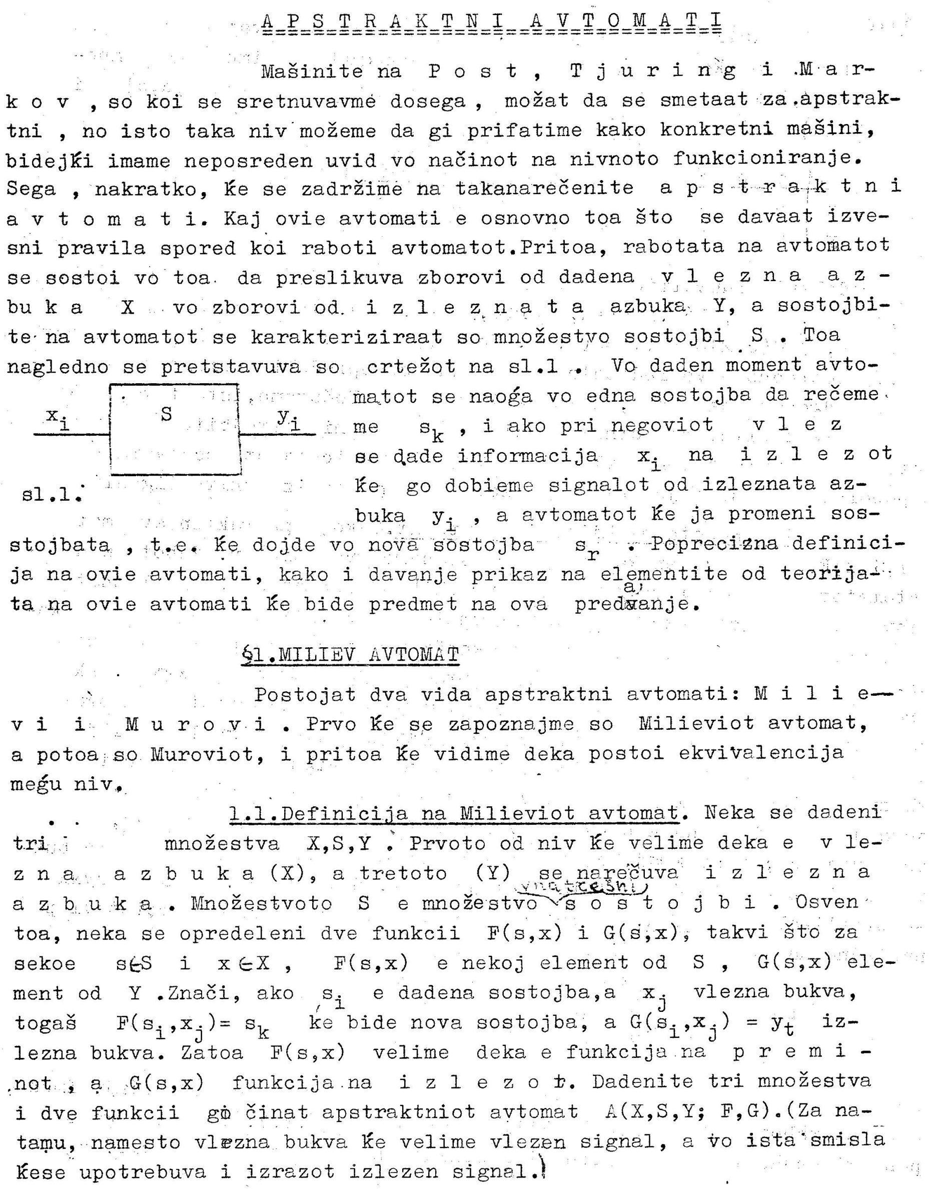
Glushkov starts with the concept of algorithms in terms of machines by Post, Turing, and Markov, as well as knowledge on propositional calculus. Basing on abstract automata the central topic in Cybernetics for Glushkov is the concept of self-organizing system. The knowledge of computers and programming is necessary, and ALGOL is covered in the book. Finally the book covers the predicate calculus and automated reasoning; which is a topic of classical Artificial Intelligence, although Glushkov does not use that term. The book gave algebraic treatment of most of the topics covered. The book brought ideas from other researchers in the field, for example from Rosenblatt and his Perceptron architecture [24][25] and Selfridge and his Pandemonium architecture [26]. It contained also the newest research of Glushkov himself on abstract automata and self organizing systems [16].

4. 1968: The First Seminar on Cybernetics in Macedonia

Having a vision of the development of mathematics, computer science and cybernetics, and motivated by the book of Glushkov, in 1968 professor Cupona organized the First Seminar on Cybernetics in Macedonia.

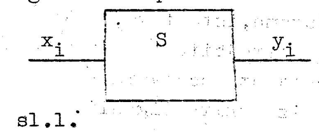
Nine lectures were presented from university professors, including the professors mentioned above and Dr Cupona himself. He also invited lecturers related to cybernetics from standpoint of control theory (Pane Vidincev) as well as from standpoint of biology (Lav Lozinski). The tenth and final lecture was assigned to a high school student, the author of this paper. At that time there was no lecturer who was working on abstract automata, and there was no lecturer in the area of computer science in Macedonia. Moreover, the chapter on automata theory from the Glushkov's book was already assigned to the student as a Matural Thesis. While the supervisor of the Matural Thesis was professor Gorica Ilieva, many discussions with Dr Cupona were on the topic. So the author delivered the lecture for which he prepared the lecture handouts [2] that were edited by Dr. Cupona (Figure 1).

We would conclude this first part of the paper pointing out the vision of Professor Cupona toward the development of mathematics, computer science, and cybernetics in Macedonia, that it should include automata theory and related topics.



A P S T R A K T N I   A V T O M A T I

Mašinite na P o s t , T j u r i n g i M a r k o v , so koi se sretnuvavme dosega , možat da se smetaat za apstraktni , no isto taka niv možeme da gi prifatime kako konkretni mašini, bidejḱi imame neposreden uvid vo načinot na nivnoto funkcioniranje. Sega , nakratko, ḱe se zadržime na takanarečenite a p s t r a k t n i a v t o m a t i. Kaj ovie avtomati e osnovno toa što se davaat izvesni pravila spored koi raboti avtomatot.Pritoa, rabotata na avtomatot se sostoi vo toa da preslikuva zborovi od dadena v l e z n a a z - bu k a X vo zborovi od i z l e z n a t a azbuka Y, a sostojbite na avtomatot se karakteriziraat so množestvo sostojbi S . Toa nagledno se pretstavuva so crtežot na sl.1 . Vo daden moment avtomatot se naoǵa vo edna sostojba da rečeme s_k , i ako pri negoviot v l e z se dade informacija x_i na i z l e z ot ḱe go dobieme signalot od izleznata azbuka y_i , a avtomatot ḱe ja promeni sostojbata , t.e. ḱe dojde vo nova sostojba s_r . Popreciзna definicija na ovie avtomati, kako i davanje prikaz na elementite od teorijata na ovie avtomati ḱe bide predmet na ova predavanje.



sl.1.

§1.MILIEV AVTOMAT

Postojat dva vida apstraktni avtomati: M i l i e v i i M u r o v i . Prvo ḱe se zapoznajme so Milieviot avtomat, a potoa so Muroviot, i pritoa ḱe vidime deka postoi ekvivalencija meǵu niv.

1.1.Definicija na Milieviot avtomat. Neka se dadeni tri množestva X,S,Y . Prvoto od niv ḱe velime deka e v l e z n a a z b u k a (X), a tretoto (Y) se narečuva i z l e z n a a z b u k a . Množestvoto S e množestvo vnatrešni s o s t o j b i . Osven toa, neka se opredeleni dve funkcii F(s,x) i G(s,x), takvi što za sekoe $s\in S$ i $x\in X$, F(s,x) e nekoj element od S , G(s,x) element od Y .Znači, ako s_i e dadena sostojba,a x_j vlezna bukva, togaš $F(s_i,x_j)= s_k$ ke bide nova sostojba, a $G(s_i,x_j) = y_t$ izlezna bukva. Zatoa F(s,x) velime deka e funkcija na p r e m i n o t , a G(s,x) funkcija na i z l e z o t. Dadenite tri množestva i dve funkcii go činat apstraktniot avtomat A(X,S,Y; F,G).(Za natamu, namesto vlezna bukva ḱe velime vlezen signal, a vo ista smisla ḱe se upotrebuva i izrazot izlezen signal.)

FIGURE 1. Beginning of the first lecture on abstract automata in Macedonia [2]. It contains a handwritten editing note by professor Cupona.

5. Introduction to Automata Theory

What follows in this chapter is description of the concept of abstract automaton and some important applications of modeling in mathematics, computer science, and cybernetics.

5.1. **Definition.** Abstract automaton is a pair of difference equations

$$s(t+1) = \delta(s(t), x(t)) \tag{5.1}$$

$$y(t) = \lambda(s(t), x(t)) \tag{5.2}$$

involving three sets X, S, and Y, and two mappings.

$$\delta : X \times S \to S$$

$$\lambda : X \times S \to Y$$

A distinct terminology (which we call abstract automata ontology) is associated with the concept of automaton:

X - set of inputs $\{x_1, x_2, \ldots, x_p\}$
Y - set of outputs $\{y_1, y_2, \ldots, y_q\}$
S - set of (internal) states $\{s_0, s_1, s_2, \ldots, s_{n-1}\}$.
δ - state-change function
λ - output computation function

We would like to stress that the concept of abstract automaton inherently contains the *concept of* *<u>state</u>*, a crucial concept in many theoretical constructs and applications. It is also important that the number of states is finite. The state from the previous time step should be remembered, as required by the difference equation (1), and requires an initial condition, such a $s(0) = s_0$. Also, a difference equation assumes discrete time. Discrete time can be defined in several ways and we propose a definition that the automaton works in discrete time if between time steps t and $t+1$ the system is not observed (or is not observable).

5.2. **Block diagram of an abstract automaton.**

The pair of difference equations (1) can be visualized as a system represented with its block diagram, as shown in Figure 2.

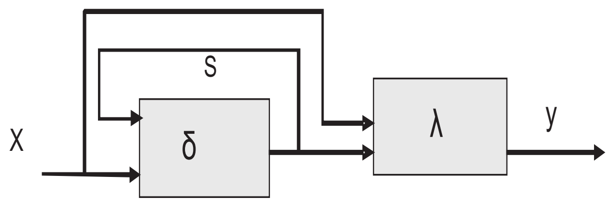


Figure 2. Block diagram of an abstract automaton

The state change function δ and the output computation function λ are in general nonlinear and are often given in tabular form. The automata are very often represented as graphs, where nodes of the graph represent states of an automaton, and graph transitions describe the state changing function δ; the output function is assigned to the graph transitions.

There are two types of automata, Mealy type and Moore type. The Mealy type automaton is described above, with functions δ and λ, and shown in Figure 2. The function λ is computed taking into account the input of the automaton and the state. In Moore type automaton the output computation function depends solely on the state of the automaton, so the second equation (5.2) has a form $y(t) = \mu(s(t))$. Often a Moore automaton is named a *state machine.*

6. Modeling with Abstract Automata

Abstract automata are convenient for modeling various types of systems. The concept of a system here is left undefined, intuitively it is an entity with inputs and outputs; states are often considered part of a system. Here we will be interested in modeling systems which are denoted as agents. They are related to the concept of environment. After presenting definition of an agent we will describe two types of agents: abstract computational machines and abstract neurons.

6.1. **Agents.**

The following system of equations describes an agent interacting with an environment.

agent:

$$s(t+1) = \delta(s(t), x(t)), \;\; s(0) = s_0 \tag{6.1}$$

$$y(t) = \lambda(s(t), x(t)) \tag{6.2}$$

environment:

$$x(t+1) = \mu(x(t), y(t)) \tag{6.3}$$

with the following ontology (agent ontology)

X - set of situations $\{x_1, x_2, \ldots, x_m\}$
Y - set of actions $\{y_1, y_2, \ldots, y_q\}$
S - set of (internal) states $\{s_0, s_1, s_2, \ldots, s_{n-1}\}$.
δ - state-change function
λ - agent action computation function
μ environment state change and new situation computing function

Note that the environment is described as a state machine and its states in form of situations, are presented to the agent.

Block diagram of an agent is given in Figure 3.

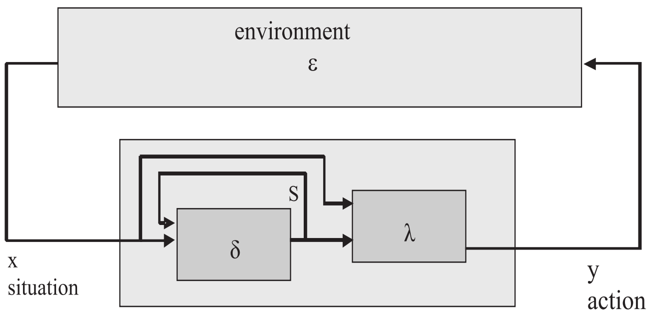


Figure 3. The concept of an agent

So the *agent can be viewed as abstract automaton interacting with its environment.*

6.2. **Abstract Computational Machines.**
An Abstract Computational Machine (also named Turing Machine or Post machine) consists of an abstract automaton and an environment represented by a (potentially infinite) tape with symbols. The following system of equations describes this type of agent

Agent: finite state automaton

$$s(t+1) = \delta(s(t), x(t)), \quad s(0) = s_0 \tag{6.4}$$

$$y(t) = \lambda(s(t), x(t)) \tag{6.5}$$

environment: potentially infinite tape:

$$i(t+1)) = \tau(i(t), y(t)), \quad i(0) = 0 \tag{6.6}$$

$$x(t) = \mu(i(t)) = x_i(t) \tag{6.7}$$

with the following Abstract Computational Machine ontology

X - tape symbols $\{x\}$

Y - output actions $\{read(x), write(x), incr(i), decr(i), halt\}$

S - set of (internal) states $\{s_0, s_1, s_2, \dots, s_{n-1}\}$.

δ - state-change function

λ - action computation function: position change function (next i)

τ - function computing position of the automaton relative to the tape

μ - function presenting new situation (tape symbol) to the agent

I infinite set of integers, tape positions; x_i is symbol at position i;

$\mathbf{x}(0) = (\dots, x_{-1}; [x_0 s_0] x_1, x_2, \dots)$, string, initial position of the computational machine

$\mathbf{x}(t) = (\dots, x_{i-1}; [x_i s] x_{i+1}, x_{i+2}, \dots)(t)$, string, current state of computational machine

The block diagram of an abstract computational machine is given in Figure 4.

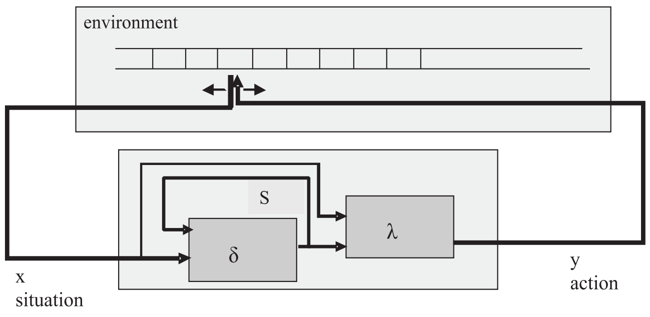


FIGURE 4. An abstract computational machine

The machine starts its computation with a tape situation described by a string $\mathbf{x}_0 = (\dots, x_{-3}, x_{-2}, x_{-1}; [x_0, s_0]; x_1, x_2, x_3, \dots)(t = 0)$ where x_0 is the tape symbol at initial position $i = 0$ of the automaton and s_0 is the initial state of the automaton. The automaton moves either to the right with its action incr(i), or to the left with its action decr(i). Here we introduce increment and decrement instead of move-right and move-left actions, since we suggest a program counter as a part of a computational machine. The computation ends when the computational machine executes its halt action.

6.3. **Abstract Neurons.**

Abstract Neuron (or Formal Neuron, or Artificial Neuron) is an abstract automaton with many inputs and one output. States are usually represented with the symbol w.
agent: abstract neuron

$$w(t+1) = \delta(w(t), x(t),), \quad w(0) = w_0 \tag{6.8}$$

$$y(t) = \lambda(w(t), x(t)) \tag{6.9}$$

environment:

$$x(t+1) = \varepsilon(y(t)) \tag{6.10}$$

X - set of input (synaptic) vectors , usually binary vectors $\mathbf{x} = [x_1, x_2, \dots, x_p]$
Y - axon output, a binary variable
W - set of (internal) states, vector of synaptic weights $\mathbf{w} = [\theta = w_0, w_1, w_2, \dots, w_n]$.
δ - state-change (learning) function, $\mathbf{w}(t) = \delta(\mathbf{w}(t-1), x(t))$
λ - action computation function,
ε environment presenting new synaptic input function

Figure 5 shows the block diagram of an abstract neuron. The thick line shows a vector.

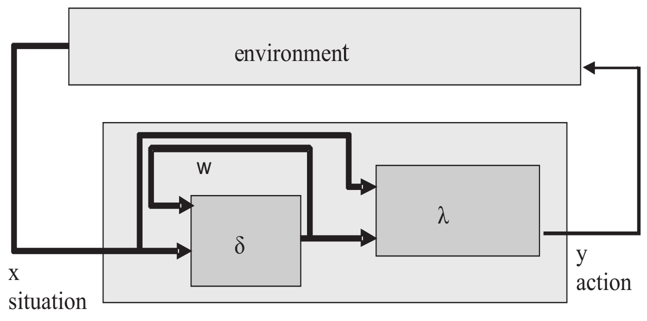


Figure 5. Abstract neuron

The state changing function is here known as learning function or synaptic plasticity function. The output function is usually computed as $\lambda : y = sgn(\mathbf{wx} - \theta)$ where $\mathbf{wx}$ is vector inner product, θ is known as neural threshold, and sgn(0) computes 1 for a nonnegative argument, otherwise gives 0. The environment contains two parts, internal and external. The internal environment consists of other neurons whose outputs are inputs to the considered neuron. The external environment gives own inputs to the neuron.

7. Cupona and the Mathematics Institute with Numeric Center

In this chapter we will mention the work of professor Cupona in forming the Mathematics Institute with Numeric Center, and the work related to abstract automata and self-organizing systems carried out at the Institute.

Professor Cupona invested his energy in building mathematics in Macedonia, especially algebra; however he was also building computer science. In addition of being active in nurturing high school talent through mathematical competitions and mathematical schools, he was engaged in building a *research infrastructure*. He invested energy and time in building the Mathematics Institute with Numeric Center. The Center started in 1967 with seminars like the above mentioned Cybernetics seminar, and eventually it became a research institution employing various types of researchers such as mathematicians, engineers, computer scientists, and economists. Here we will mention some of the researchers employed by the Institute: Simeon Ivanovski, Smilka Zdravkovska, Risto Sokarovski, Tatjana Surlezanovska, Smile Markovski, Vanco Kusaktov, Margita Kon-Popovska, Dragan Mihajlov, Dragan Nikolic, Donco Dimovski, Stevo Bozinovski, among others.

Among various directions of research, here we will mention the research influenced by Dr Cupona in directions toward automata and self-organizing systems. As result of this research the first papers on the subject were: a journal paper on psychocybernetics in 1975 [3], a technical report on formal neurons in 1976 [4], and a proceedings paper on perceptrons in 1976 [5].

In 1976 Dr Cupona published his book "Algebraic structures and real numbers" [13] which influenced his students in algebraic reasoning.

8. 1968-1982: Contribution to Automata Theory and Theory of Self Organizing Sytems

This chapter covers the results in area of automata and self-organizing systems of the author obtained in contact with Dr Cupona in the period from 1968 till 1982. Here we will describe the research in two types of automata: teachable automata (perceptrons) and state self-evaluation automata (crossbar adaptive arrays).

8.1. **Teachable automata: Perceptrons.**

The neural networks research started 1943 with the first mathematical model of a neuron [21]. Variations of the original model were proposed [e.g. 22]. The model includes synapses, internal potential, activation (firing) threshold, and single output, axon. Potential learning mechanism was proposed in 1950 [18] that the learning takes place in modification of synapses (synaptic weights). The neural network named perceptron proposed by Rosenblatt [24][25] includes the synaptic weights in learning for pattern classification. The perceptron studied by Glushkov contained maximum selector mechanism for choosing actions, proposed by Selfridge [26]. Learning became a focus of several researchers at that time, and we will mention only two, Stenbuch with his learning matrix [27] and Nilsson with his learning machines [23]. Most of the effort was given on the learning process, with concepts such a patterns, synaptic weights, synaptic plasticity, and weight space [e.g. 14]. Our work followed the Glushkov's direction toward the teaching aspects of the perceptrons. We will first describe the perceptron in terms of abstract automaton.

perceptron: pattern classifier, learner

$$w(t+1) = \delta(w(t), (x(t), u(t), r(t))), \quad w(0) = w_0 \tag{8.1}$$

$$y(t) = \kappa(C(t)) = \lambda(w(t), x(t), \theta) \tag{8.2}$$

environment: teacher (or instructor, or trainer, or advisor, or supervisor)

$$(r(t), u_i(t)) = \varepsilon(x(t), y(t)) \tag{8.3}$$

$$x(t+1) = \tau(x(t), y(t)) \tag{8.4}$$

Perceptron ontology:

X - pattern (situation) set, $\{\mathbf{x}_1, \dots, \mathbf{x}_j, \dots \mathbf{x}_m\}$ usually m binary d-dimensional row vectors

C class set $\{C_1, C_2, \dots, C_n\}$, n classes ($n \le m$) to which patterns are classified

Y - action set, $\{y_1, y_2, \dots, y_n, y_0\}$, actions corresponding to classes, y_0 is "unknown" class

W - set of states, a matrix of n row vectors $\mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_n]^T, \dim(\mathbf{w}) = d$

r teacher's evaluation of the learner, $r = 0$ if learner classified correctly, else $r = 1$

U teacher's advice, u_i if the shown pattern $\mathbf{x}$ belongs to the class C_i

δ - state-change function, $\mathbf{w}_i(t+1) = \mathbf{w}_i(t) + r(t+1)u_i(t)\mathbf{x}(t)),\ r(t+1) = 1$ if $\mathbf{x}$ belongs to y_i

λ - behavior computation function, $y = y_i$ if $\mathbf{w}_i\mathbf{x} - \theta_i > \mathbf{w}_k\mathbf{x} - \theta_k$, for all $k \neq i$, else y_0

θ is neural threshold

ε teacher's evaluation-of-the-learner function

σ teaching strategy function, which next lecture (pattern) to present

The block diagram is shown in Figure 6.

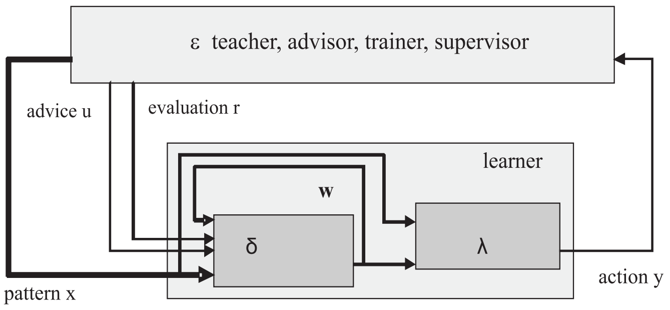


FIGURE 6. Perceptron, a trainable pattern classification automaton

Let us define *behavior* as sequence of actions. If a sequence repeats itself, let us call it *behavioral routine*. The behavioral routine of the teacher in case of training the perceptron to classify patterns into classes is

```
repeat
    perform examination trial ($r = 0$):
        chose pattern $x$, show to learner, receive learner's response $y$;
    if the response is incorrect,
        perform teaching trial ($r = 1$) : give correct advice $u$;
    else continue
until stopcondition
```

The objective of the teaching process is: Given a learner L (perceptron). Given set $X = \{x_1, \ldots, x_j, \ldots x_m\}$ of m patterns (or lectures, or situations). The learner performs classification K of the given set of patterns into set of $n+1$ ($n \leq m$) behaviors $\{y_1, \ldots, y_i, \ldots y_n, y_o\}$ where y_0 is the "don't know" behavior. Given correct classification (partition) K_c of the set X into n classes, represented by the learner's behaviors. Design an iterative procedure T (teacher) which will implement an evaluation function ε of the behavior of L and a teaching strategy σ, such that the procedure T assures convergence of K toward K_c.

The following is our paradigm shift statement, a generalization of [5]: Given a perceptron with n neurons represented by weight vectors $\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_n$ (real weight space). Given m patterns $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m,\ n \leq m$. The learning process for the perceptron can be represented in an integer space $\mathbf{P}$ (teaching space), as integer programming problem:

$$\text{find} \qquad \min(\mathbf{1p}) \tag{8.5}$$

$$\text{satisfying :} \;\; \mathbf{Ap} > \mathbf{q} \tag{8.6}$$

where vector $\mathbf{1}$ has all components "1", $\mathbf{1p}$ is inner product $(= p_1+p_2+\ldots p_m,)$, $\mathbf{A}$ is matrix containing inner products (similarities) between patters, $\mathbf{p}$ is the curriculum vector showing how many times p_j each pattern $\mathbf{x}_j$ appears in a teaching trial, and $\mathbf{q}$ is vector of initial conditions, transfer of a previous training.

Let us note that in the system of inequalities (8) there are no weights, vectors $\mathbf{w}$. The initial values of the weights $\mathbf{w}_0$, are part of the transfer of training effect, vector q. If there is no previous transfer of training, the relation (8.6) becomes $\mathbf{Ap} > \mathbf{0}$. In such a case what matters are inter-pattern similarities rather than patterns themselves. So, while the classical learning problem [14] searches for vectors $\mathbf{w}_1^*, \mathbf{w}_2^*, \ldots, \mathbf{w}_n^*$ for which the learning process converges, the teaching approach searches teaching curriculum $\mathbf{p}$ of minimal length. Let us also note that vector $\mathbf{p}$ and matrix $\mathbf{A}$ are observable in a teaching process, while vectors $\mathbf{w}$ (learner's memory) is not observable in biological learners. Transfer of training concept [15] is also part of the paradigm shift statement.

Part of the research in teaching systems theory was submitted in 1981 to an IEEE Transactions journal and later published [10]. It was an example of Dr Cupona's contribution to motivation of researchers from Macedonia in the area of computer science and cybernetics to publish in high level journals.

8.2. **Self-evaluation automata: Crossbar Adaptive Array.**

Here we will describe our contribution to automata theory with designing an automaton with ability to self-evaluate its internal state. The automaton here is described with an additional, third equation. However, note that the equations (8.7) and (8.8) can be written as a composite function, such that the automaton is described by two equations. The following automaton named Crossbar Adaptive Array (CAA) was proposed in 1981 [6][7]:

self-evaluation automaton

$$w(t+1) = \delta(w(t), v(t+1)), \quad w(0) = w_0 \tag{8.7}$$

$$v(t) = \nu(w(t), x(t)) \tag{8.8}$$

$$y(t) = \lambda(w(t), x(t)) \tag{8.9}$$

environment: a graph, provider of the next situation

$$x(t+1) = \varepsilon(x(t), y(t)) \tag{8.10}$$

self-evaluation automaton ontology:
X - situation set, usually m nodes of a graph, $x_1, \ldots, x_j, \ldots x_m$
Y - action set, $y_1, y_2, \ldots, y_n,$, n transitions from a graph node.
W - set of (internal) states, m column vectors of the matrix $\mathbf{W} = [\mathbf{w}_1, \ldots, \mathbf{w}_j, \ldots, \mathbf{w}_m] = [w_{ij}]_{i=1,\ldots,n; j=1,\ldots,m}$

ν state evaluation function, $\nu(t) = sign(\max_{j=1,\dots,m}(w_j(t))$, where $sign(\ )$ is a function that gives 1 for positive argument, -1 for negative, and 0 for zero argument.

δ - state-change function, $w_{ij}(t+1) = w_{ij}(t) + v_k(t+1)$, if action i in node j gives node k.

λ - action computation function, given x_j, $y = y_i$ $w_{ij}x_j = \max_{b=1,\dots,n}(w_{bj})$, else random

ε environment responding function, agent's action y_i in node x_j gives node x_k.

Figure 7 shows the block diagram of a state evaluation automaton. It is named Crossbar Adaptive Array (CAA) because the memory array computes, in a crossbar fashion, both the actions and the state evaluations. The overall state evaluation is named feeling (emotion) of being in the current state. As Figure 7 suggests emotion can also be used as action if it is displayed to the environment. However here we only consider automaton that computes its state evaluation and uses it in the learning (memory update) process.

Important feature of the CAA architecture is that the only input it receives from the environment is current situation. No advice and no reinforcement is received. The CAA architecture has feature of a initial (genetic) vector that corresponds to the environment and gives inherited evaluation of some environment situations it encounters [8][9].

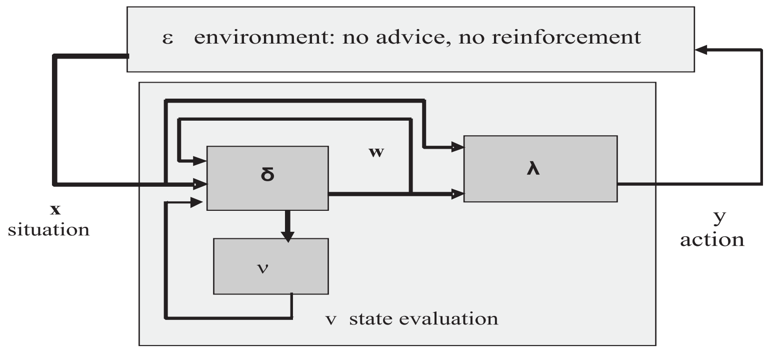


FIGURE 7. The state self-evaluation automaton

For the automaton in Figure 7 the behavioral routine is defined as:

```
    $\lambda$ : produce random action $y$
repeat :
    $\delta$: receive situation $x_j$
    $\lambda$: produce action $y_i$
    $\varepsilon$ : provide new situation $x_k$
    $\nu$ : evaluate $x_k$, obtain $v_k = \nu(x_k)$
    $\delta$: learn, update memory, $w_{ij} + v_k$
until stop condition
```

By introducing states self-evaluation, a need appeared of representation of state evaluation in an automaton graph. It was chosen that emoticons should be used as representation of a state value. Automaton with state evaluations using emoticons is named *emotional automaton.* Figure 8 shows examples of problems represented using graphs of emotional automata. Emotional automata are also convenient in representing language acceptors.

The CAA architecture was used in solving some of challenging problems for neural networks and learning theory. A general challenging problem was the problem of learning with delayed reinforcement (or learning with delayed reward) [20]. Here we give a short description of the problem: How to perform learning if after series of actions by the agent there is no response from the environment neither in form of evaluation (reward) nor in form of advice? An instance of this general problem is the maze learning problem: Given a graph that gives evaluation (reward) only at the goal state; an agent at start state should learn a path toward the goal state. Another instance is the pole balancing learning problem: Given inverted pendulum system (cart and pole) with known dynamics [e.g. 12]; the angle and angular velocity of the inverted pendulum is known in each step; the evaluation (actually punishment) only comes if the pendulum falls down. Learn a control policy that will keep the pendulum upwards. Figure 8 shows both the instances of the general delayed reinforcement learning problem, using emotional graphs.

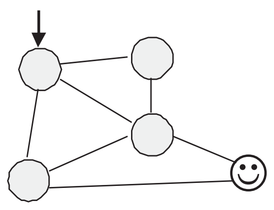


Fig. 8a.

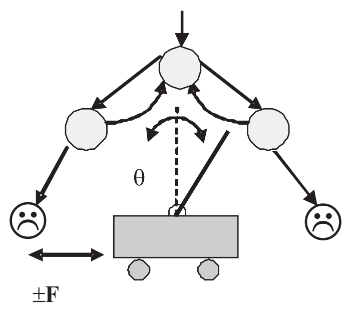


Fig. 8b.

FIGURE 8. Emotional automata graphs representing instances of delayed reinforcement learning problem Fig 8a. Maze learning problem. Fig. 8b Inverted pendulum balancing learning problem

The delayed reinforcement learning was considered and solved while the author was working with the Adaptive Networks (ANW) Group of the University of Massachusetts at Amherst, during 1980-1981. At that time members of the group were Nico Spinelli (Principal Investigator), Michael Arbib, Andy Barto, Rich Sutton, Charles Anderson, Jack Portefield, and Stevo Bozinovski. The ANW group was funded by a grant from Wright Patterson Air Force Base in Dayton, Ohio. Project officer was Harry Klopf, with his leading ideas that we should design goal seeking systems from goal seeking components. The challenge of solving the delayed reinforcement learning problem, and the maze running instance, was stated by Rich

Sutton. The pole balancing instance of the problem was stated by Charles Anderson. Solution of both instances of the problem was achieved in 1981 [6][7]. First paper was published in 1982 [8][9] and a more comprehensive one was published later [11]. The crucial innovation was the state self-evaluation (feeling, emotion) concept. The state evaluation and state value turned out to be known concepts in Dynamic Programming research [1][19]. However the state self-evaluation and feeling were new concepts added to the neural networks and learning systems research.

As conclusion to this chapter, the directions provided by Dr Cupona, and knowledge gained from the Glushkov's book, enabled a graduate student from Macedonia to solve a challenging problem considered by the ANW group that in 1981 carried out a leading research in neural networks and self-organizing systems.

9. Conclusion

This work is a report of an example of influence of professor Cupona on generation of students, from high school level to PhD level. In the example described here, professor Cupona selected a high school student who was active in mathematics competitions and gave him a task to read a chapter on automata theory from a relevant, just published book containing up to date research on a subject of abstract automata and self-organizing systems. He further encouraged the student to give a lecture on automata theory on a seminar on which all other lectures were delivered by university professors. The student continued in the same direction and did both his master's and doctoral theses on the subject. In the process he published a paper in a journal from the series IEEE Transactions. Having acquired the needed knowledge the student was able to solve a difficult problem in neural networks, the delayed reinforcement learning problem, which was stated as challenge in front of the ANW group from the University of Massachusetts at Amherst in 1981. This is just one of examples of influence of professor Cupona upon younger generation and development of mathematics, computer science, and cybernetics in Macedonia and worldwide.

In retrospective, we can say that the first seminar on Cybernetics that professor Cupona organized in 1968 was the single event that started the computer science development in Macedonia. In contemporary computer science, automata theory provides fundamentals for digital hardware (flip-flops and logic gates) as well as for software (programs). Back in 1968 Dr Cupona had a vision toward right direction.

References

[1] R. Bellman, *Dynamic Programming*, Princeton University Press, 1957.
[2] S. Bozinovski, *Abstract automata (edited by G. Cupona) Topic 10. First Seminar on Cybernetics*, Mathematical Institute with Computing Center, Skopje, 1968.
[3] S. Bozinovski, *Introduction to psychocybernetics* (in Croatian), Psychological Review 5(1-2): 115-128, Zagreb, 1975.
[4] S. Bozinovski, *An approach toward threshold elements and formal neurons*, (In Macedonian), Mathematics Institute with Numeric Center, Skopje, 1976.

[5] S. Bozinovski, A. Fulgosi, *Influence of pattern similarity and transfer of training on training the base perceptron B2* (in Croatian) Proc Symp. Informatika, 3-12-1-5, Bled, 1976.

[6] S. Bozinovski, *A self learning system using secondary reinforcement.* First report on CAA architecture, Computer and Information Science (COINS) Department, University of Massachusetts at Amherst, November 25, 1981.

[7] S Bozinovski, *Inverted pendulum learning control*, Adaptive Networks Group (ANW) Memo, COINS Department, University of Massachusetts at Amherst, December 10, 1981.

[8] S. Bozinovski, *A self-learning system using secondary reinforcement.* Abstract 87, Published Abstracts of the Sixth European Meeting on Cybernetics and Systems, Vienna, April 1982.

[9] S. Bozinovski, *A self-learning system using secondary reinforcement*, in E. Trappl (ed.) Cybernetics and Systems Research, North Holland, 1982, p. 397-402.

[10] S. Bozinovski, *A representation theorem of linear pattern classifier training*, IEEE Transactions on Systems, Man, and Cybernetics 15: 159-161, 1985.

[11] S. Bozinovski, *Crossbar Adaptive Array: The First Connectionist Network that Solved the Delayed Reinforcement Learning Problem.* In: A. Dobnikar, N. Steele, D. Pearson, R. Alberts (eds.) Artificial Neural Networks and Genetic Algorithms, Springer Verlag 1999, p. 320-325.

[12] R. Cannon, *Dynamics of Physical Systems*, McGraw Hill, 1967.

[13] G. Cupona, *Algebraic Structures and Real Numbers* (In Macedonian), Prosvetno Delo Publisher, 1976.

[14] R. Duda, P. Hart, *Pattern Recognition and Scene Analysis*, Willey Interscience, 1973.

[15] R. Gagne, K. Baker, H. Foster, *On the relation between similarity and transfer of training in the training of discriminative motor tasks*, The Psychological Review 57:67-79, 1950.

[16] V. Glushkov, *Self organizing systems and abstract theory of automata.* (In Russian) Zhurnal Vychislitelnoi Matematiki i Matematicheskoi Fiziki 3: 459-466, 1962.

[17] V Glushkov, *Introduction to Cybernetics*, Academic Press, 1966.

[18] D. Hebb, *The Organization of Behavior*, Wiley, 1949.

[19] R. Howard, *Dynamic Programming and Markov Processes*, MIT Press, 1960.

[20] F. Keller, R. Schoenfeld, *Principles of Psychology*, Appleton-Century-Croffts, 1950.

[21] W. McCulloch, Pitts, *A logical calculus of the ideas immanent in nervous activity*, Bulletin of Mathematical Biophysics 5: 115-133, 1943.

[22] M. Minsky, *Finite and Infinite Machines*, Prentice Hall, 1967.

[23] N. Nilsson, *Learning Machines*, McGraw-Hill, 1965.

[24] F. Rosenblatt, *The perceptron: A probabilistic model for information storage and organization in the brain*, Psychological Review 65: 386-408, 1958.

[25] F. Rosenblatt, *Principles of Neurodynamics*, Spartan Book, 1962.

[26] O. Selfridge, *Pandemonium: A paradigm for learning*, Proc Symposium on Mechanization of Thought Process, London, Her Majesty Stationary Office, p. 523-526, 1958.

[27] W. Steinbuck, *The learning matrix* (in German) Kybernetik 1:36-46, 1961.

Department of Mathematics and Computer Science,
University of South Carolina, Orangeburg, SC, USA
E-mail address: `sbozinovski@scsu.edu`