COMPUTER SCIENCE EDUCATION 7, 2 (1996)

Cognitive-Conceptual Model for Integrating Robotics and Control Into the Curriculum

David Mioduser
Ilya Levin
Tel-Aviv University

The phenomenon of control is an essential component of our everyday natural, social and artificial environment. Control-related concepts have become a central component of many core topics in modern technology education. However, evidence already collected shows that students have serious difficulties in transcending the phenomenal or behavioral understanding of a system's functioning, towards more formal definitions of the control process. In this paper we suggest a framework for the design of learning environments of control concepts and design skills. Two main paradigms are suggested as the conveyors of very different cognitive approaches to control: *programming* or software oriented and *design* or hardware oriented paradigms. The control of a Lego robot is used to elaborate on the different paradigms, and an example of computer implementation of control using spreadsheet software -the spreadsheet-book metaphor for representing control- is presented in detail.

1. INTRODUCTION

The phenomenon of control is an essential component of our everyday natural, social and artificial environment [1,2], and control-related concepts have become a central topic in modern technology education [3]. Computers, programmable controllers, CNC, CAM, dynamic systems and other important subjects of the technology curriculum are based on the general idea of control. These concepts and skills pervade now all levels of technology education, becoming a central component in both "Technological Literacy" and specialization curricula [4].

Evidence already collected shows that students have serious difficulties in transcending the phenomenal or behavioral understanding of a system's functioning, towards more formal definitions of the control process [5,6,7,8]. Teaching control (analysis and design) concepts implies facing several key questions, such as:

The present work is being done at the Knowledge Technology Lab of the Science and Technology Education Center at the Tel Aviv University. The authors wish to thank LEGO-DACTA for their technical advice and software support allowing us to connect the LEGO controller via the different interfaces under development at our Lab.

Correspondence and requests for returns should be sent to David Mioduser, Tel-Aviv University School of Education Ramat-Aviv, 69978; ISRAEL, FAX#: 972-3-6409477, e-mail: miodu@ccsg.tau.ac.il.

- What model of control will be appropriate to teach to students seeking different learning goals (e.g., high school students learning technological literacy, college students acquiring expertise, technicians being trained to repair a device)?
- How do different paradigms for defining and representing control affect the design and development of curricular solutions (e.g., learning environments, instructional sequences)?

In this paper we propose a way to start dealing with these and related questions, based on a framework we have elaborated for teaching control concepts and skills [9]. A key component in this framework, the "Multiple-constructs Framework for Control", is the characterization of different approaches or paradigms for defining and implementing control. These different paradigms lead to the design of different teaching strategies, and present to the students different cognitive demands. One of our currents curricular efforts is aimed at implementing learning environments for the different paradigms based on the use of computer spreadsheet software (Excel). In the following we present the conceptual framework and an example of the spreadsheet implementation of the control of a Lego robot.

2. CONTROL SYSTEMS

Let us start with the description of a controlled system we have adopted in our framework. It is based on the principle of dividing the system into two main components: *operating unit* (OU) and *control unit* (CU). The operating unit contains performing elements. The OU is in fact the physical device including structural parts and mechanisms, as well as all required sensors (e.g., light, touch, temperature) and effectors (e.g., motors, lights). The control unit implements the algorithm of the system's behavior.

Figure 1 shows the schematic structure of a controlled system. The set $X=\{x_1,\ldots x_L\}$ of binary signals, transferred from the OU to the CU can be named the *world state* of the system. The set of binary signals $Y=\{y_1,\ldots,y_N\}$ sent by the CU to the OU is the *set of microoperations* affecting the OU's behavior. The goal of the CU

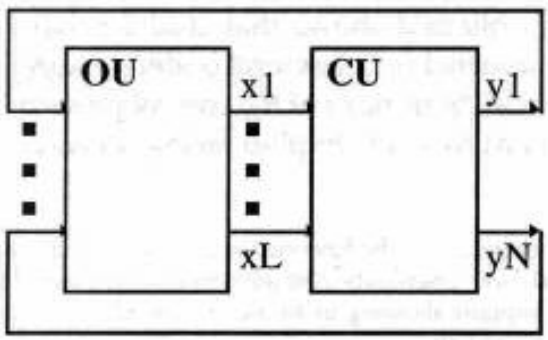

Figure 1. Representation of a General Control System.

is the generation of a sequence of signals Y, distributed in time. The functioning of the OU is dictated by this sequence.

Many examples of devices and systems that can be viewed as instances of the above presented structure are part of our immediate physical environment. A particularly rich curricular solution for bringing instances of such systems into the classroom is nowadays being introduced in the form of building and programming kits (e.g., Lego-Technic and Control Lab kits). These kits allow the students to build physical devices by means of modular building bricks, and to write computer programs to control the functioning of these devices [5]. The brick structures can be defined as the operating units of the system, while the computer program can be viewed as the implementation of the system's control unit.

These building and programming kits offer a unique opportunity for teaching and learning control concepts. But for this potential to be realized their implementation should be supported both at the cognitive and the curricular levels. This includes the development of appropriate instructional materials and learning environments. As a first step in this direction we defined a *Multiple-constructs Framework for Control,* described in the following section.

3. DEFINING AND REPRESENTING CONTROL

The instructional framework we have developed consists of two main components: the *process* component and the *representational* component. The first relates to the stages in the process of defining and implementing control. The second is the repertoire of different constructs used for defining and implementing control.

Regarding the control design process we distinguish three stages: the initial description of the (observed or desired) system's functioning, the translation of this description into formal notation or model (e.g., flow chart, state graph), and its actual implementation for controlling the system.

The second component of our framework refers to the alternative constructs we may use for representing control. The control research and development field offers a rich repertoire of notation systems [10,11]. Some of these constructs are everyday working-tools for engineers, designers and programmers dealing with control-related tasks. However, for learning purposes we found it necessary to rearrange and reorganize these disciplinary knowledge and tools. In our model we have chosen a particular set of constructs which we considered the more relevant for our purpose, and we arranged these constructs as shown in Figure 2.

We establish the distinction between two basic paradigms: *programming* (or software oriented) and *design* (or hardware oriented). Each paradigm is refereed to in the model at three levels: the conceptual approach used for defining control, its formal model or representational notation, and commonly used implementation means. This framework will be detailed in the following sections using as an example the synthesis of the control unit for a LEGO robot. The robot's performance may be described in the following way:

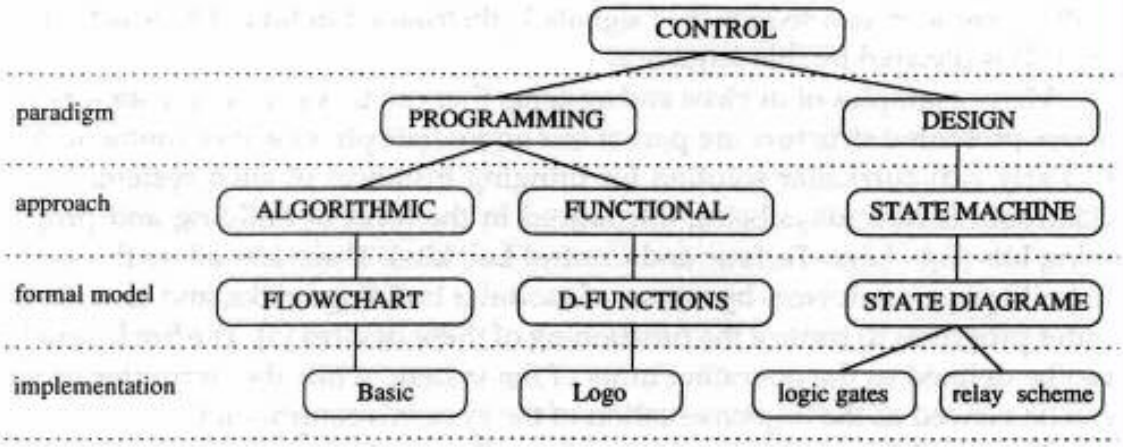

Figure 2. Mutiple-Constructs Framework for Control Definition and Implementation

It goes back and forth between two parallel walls in a path perpendicular to the walls. Whenever it touches a wall it reverses its direction.

Input signals of the control unit are: $X=\{x1 \ldots x3\}$
Output signals (microoperations): $Y=\{y1 \ldots y4\}$.
The meaning of these signals is:

$x1$ - sensor BACKWARDS on?
$x2$ - sensor FORWARDS on?
$x3$ - TOTAL STOP switch of the system on?

The control unit sends signals, which control the functioning of the operating unit. These signals are:

$y1$ - motor FORWARD on.
$y2$ - motor BACKWARD on.
$y3$ - motor FORWARD off.
$y4$ - motor BACKWARD off.
$y0$ - empty microoperation (no action in the operating unit)

3.1 The Programming (Software Oriented) Paradigm

By the programming paradigm the student assumes the existence of a control performer (e.g., a microprocessor) in charge of running the control specifications. This kind of CU can be defined as a *programmable controller*. By this paradigm, to create control means to create an appropriate program.

Within the programming paradigm, several approaches can be taken (e.g., algorithmic, functional). For example, let us present the algorithmic definition of the control unit for our *car-between-walls* example. The key formal construct for the algorithmic paradigm is the flow chart.

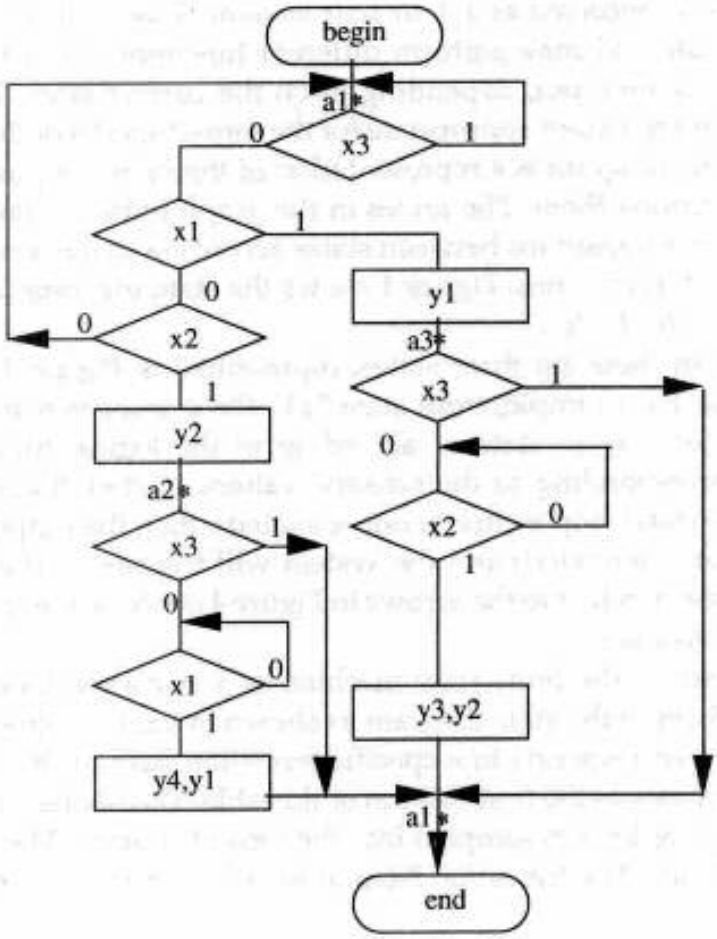

Figure 3. Flowchart for the Car-Between-Walls Device

The flow chart for our example is shown in Figure 3. In the beginning the values of both sensors, namely the forward and backward sensors, are checked. For example, if the value of the backward sensor is 1 (X1=1 in Figure 1), then the car will start moving forward according to the corresponding output signal or microoperation Y1. The car will continue its moving forward until the value of the forward sensor (X2) will equal 1. After that the forward motor will be turned off and the backward motor turned on (microoperations Y3, Y2 in Figure 1). The direction of the motors will continue to change depending on the sensors' input values (being either 0 or 1).

3.2 The Design (Hardware Oriented) Paradigm

By following the design paradigm the student focuses on the logical scheme inside the CU, implemented by means of logical elements of varied nature (e.g., logical gates, contacts, programmable logical devices). By this paradigm, to create control means to design the configuration of elements most appropriate for generating the desired behavior.

The system is conceived as a *finite state machine* (FSM). The CU can be characterized by its state and may perform different functions (i.e., changing to other states) with the same input, depending upon the current state. The formal construct we consider the more appropriate for the formal-model definition is the *state diagram*. The state diagram is a representation of the system's possible states and the transitions among them. The nodes in the graph indicate states (a_1, a_2,...a_M), and the arrows the transitions between states according to the input values which would cause such transitions. Figure 4 shows the state diagram for our example, the *car-between-walls* device.

In our system there are three states, represented in Figure 4 by three corresponding nodes. For example, from state "a1" the system may either transfer to states *a*2 or *a*3 or stay in state *a1* according to the logical function of the set $X=\{x1,x2,x3\}$ corresponding to the sensors' values. If $x1=1$ (backward sensor is "on"), and $x3=0$ (total stop switch is not activated), then the output signal will be *y*1 (forward motor activated) and the system will transfer to state *a*3. In similar way, all functions attached to the arrows in Figure 4 describe the specific transition rules from state to state.

We will describe the finite state machine as a transition table of the control unit, a tabular form of the state diagram as shown in the transition Table 1. Every row of this table corresponds to a specific transition path of the automaton. The initial state *ai* appears in the first column of the table. The Boolean function *X*(*ai*,*aj*) for the logical variables *X* is sampled into the second column. The state *aj* appears in the third column. The transition from *ai* to *aj* is performed if function *X*(*ai*,*aj*)

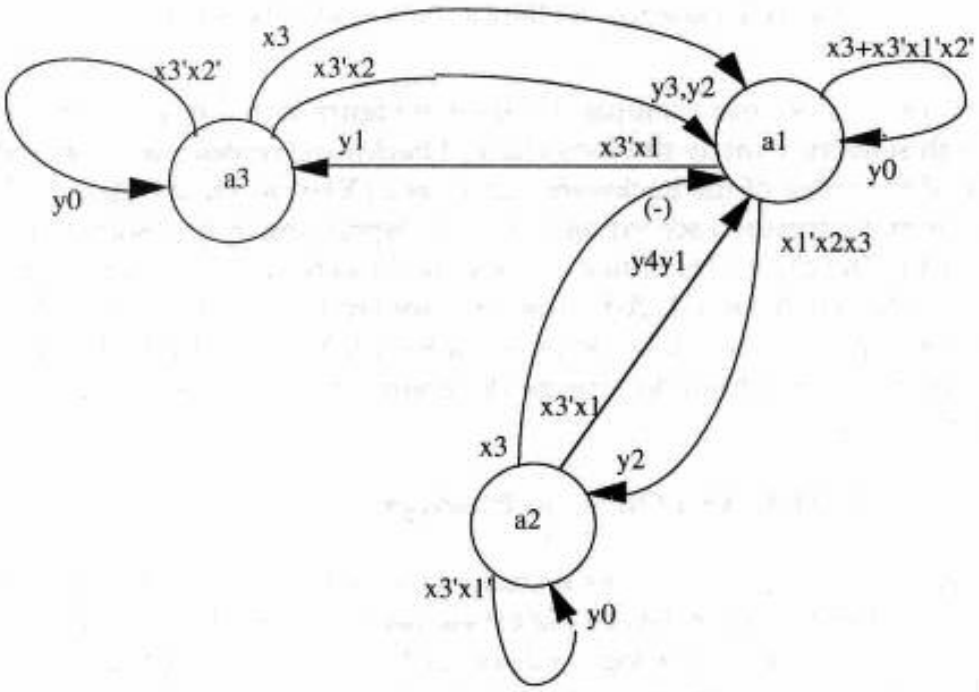

Figure 4. State Diagram for the Lego Robot

Table 1
Transition Table for the Car-Between-Walls Device

ai	*X(ai,aj)*	*aj*	*Y (ai, aj)*
a1	*x3`x1`x2`*	*a1*	*y0*
	x3`x1`x2`	*a2*	*y2*
	x3`x1	*a3*	*y1*
	x3	*a1*	*y0*
a2	*x3`x1*	*a1*	*y1, y4*
	x3`x1`	*a2*	*y0*
	x3	*a1*	*y0*
a3	*x3`x2*	*a1*	*y2, y3*
	x3`x2`	*a3*	*y0*
	x3	*a1*	*y0*

equals 1. The subset of microoperations which equals **1** on this transition is sampled in the fourth column. The complete transition table of the finite state machine for our *car-between-walls* example is shown in Table 1. Every row describes one transition. For example the sixth row shows the transition from the state $a2$ to the state $a1$, given that for the set of input variables the function $x3'x1'$ equals 1.

4. AN EXAMPLE: SPREADSHEET IMPLEMENTATION OF THE FINITE STATE MACHINE MODEL OF THE LEGO ROBOT CONTROL

In the previous sections we exemplified how control is defined and formally represented by the different paradigms. Here we will focus on the implementation of the formal model. For example, a common implementation of the algorithmic approach could be a program written in Basic or similar programming languages. For the functional approach we may use a functional programming language for implementing control. In such a language, all procedures are well-defined functions of their arguments. The most popular example of a functional programming language in education is Logo. Finally, the finite state machine can be implemented in several different forms: Hardware implementation (e.g., logical gates, PLA, ROM) or software implementation. Modern software environments give an opportunity to test the different forms of implementation of the control model. In this paper we focus on using the Excel spreadsheet program to implement the finite state machine model of the Lego robot control.

4.1 Matrix Implementation of the Control Model

The use of the spreadsheet for implementing the of the FSM, follows the *matrix model* of logical simulator proposed by Levin [12]. The schematic description of the control unit in Figure 5 will serve as the basis for defining the matrix implementation of the FSM.

In the Figure 'MC' represents an 'AND matrix' which forms the terms E1,...,EH, corresponding to transitions of the FSM; 'MD' represents an 'OR matrix'

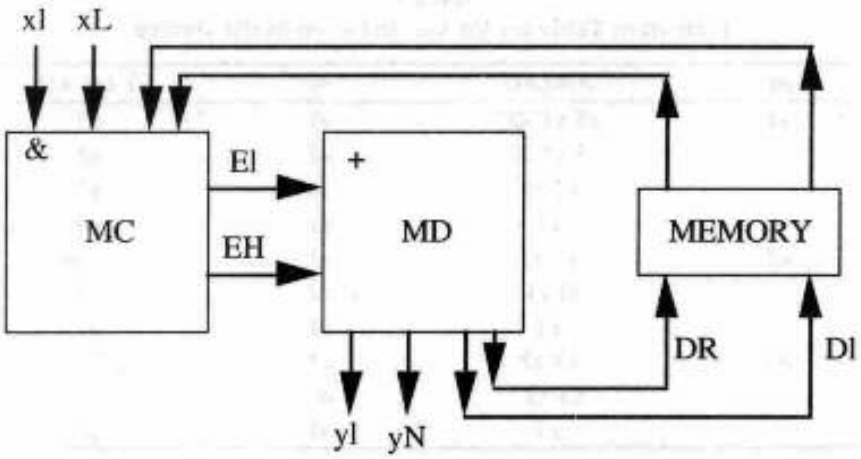

Figure 5. Schematic Representation of the Control Unit

implementing microoperations and memory functions as disjunctions of correspondent terms from the set $E1,\ldots,EH$. "Memory" represents the register for internal states store.

At any given state a_m $(m,\ldots,M)$, the FSM works as a logical scheme (without memory). Thus one can say that the FSM is a collection of different logical schemes, each corresponding to a certain state. In our example we have three logical schemes, implementing the FSM transition table (Table. 1)

Matrix descriptions of the logical scheme are usually represented in the form of tables, as the one shown in Table 2 for our car-between-walls example. The columns in such tables are marked by variables $x_1,\ldots x_L$ and functions $y_1,\ldots y_N$ or signals of states $a1,\ldots a_M$. Symbol *1* appears at the intersection of row l and column h, if the variable x_l is contained in the term X_h in the direct form. Symbol 0 appears at the intersection of row l and column h, if the variable x_l is contained in the term X_h in the negative form. Symbol # appears at the intersection of row l and column

Table 2
Matrixes for the Car-Between-Walls Device

	X1	*X2*	*X3*	*Y1*	*Y2*	*Y3*	*Y4*
	0	0	0	*	*	*	*
	0	1	0	*	1	*	*
a1	1	#	0	1	*	*	*
	#	#	1	*	*	*	*
	1	#	0	1	*	*	1
a2	0	#	0	*	*	*	*
	#	#	1	*	*	*	*
	#	1	0	*	1	1	*
a3	#	0	0	*	*	*	*
	#	#	1	*	*	*	*

h if the variable x_l is absent in the term X_h. Character *1* appears at the intersection of row X_h and column y_n or a_m if term X_h is contained in the function y_n (n=1...N) or a_m (m=1,...M), and character * in the opposite case.

This representation closely resembles a two level matrix representation of an integral circuit, having a fixed number of inputs L, conjunctions H, and disjunctions N. This matrix implementation of the circuit is called Programmable Logic Array, or PLA. Thus we will assume that the PLA implements the system of logical functions of our interest.

4.2 Spreadsheet-Based Matrix Implementation

Let us employ the matrix for simulating every logical scheme within the spreadsheet. The suggested method of simulation involves two main ideas. First, we will construct the spreadsheet-model of the system of logical functions which represents the transition table of the system. Second, we proceed to construct the spreadsheet model of the PLA making every intersection point of the matrix structure correspond with a cell of the spreadsheet-model.

We use two spreadsheets M_1 and M_2 for these purposes. M_1 is a spreadsheet-model of the logical scheme, M_2 is the spreadsheet-model of the PLA. Every cell of M_2 is programmed by a universal spreadsheet function. The function's value depends on the value of the corresponding cell of the spreadsheet-model M_1.

In accordance with the matrix structure any cell of the spreadsheet-model of the matrix of conjunctions can implement one of three different functions: reference to the contents of the left hand cell, conjunction of the variable and contents of left hand cell, conjunction of the inversion of the variable and contents of left hand cell. The spreadsheet-model of the disjunction matrix can implement one of two functions: reference to the value of the higher preceding cell or the disjunction of the higher preceding cell and corresponding output of the matrix of conjunction.

Thus, the spreadsheet implementation of the logical scheme is actually the simulation of the Programmable Logical Array. We have several matrixes, implementing logical schemes, corresponding to every state. (In our example - 3 matrixes). The only thing left now is to run this system of matrixes. For this purpose the "book" approach is suggested and described in the following section.

4.3 The "Spreadsheet-Book" Metaphor as Interface For Teaching Control

In previous work [12] the principles of logical scheme implementation in the spreadsheet have been suggested. The metaphor "sheet-logical scheme" was used there. We suggested that the functioning of the logical scheme is independent of time: the output instruction is defined only according to the value of the input variables (sensors).

However, the Control Unit not only comprises the logical scheme. The Control Unit is actually a sequential automaton. According to this view, it's functioning depends not only upon the external inputs but also upon time, or internal state of the system. The automaton is thus the set of logical schemes describing the behav-

ior of the system at every different state. Each logical scheme in the set (for each state) includes the generation of both output signals and the next-state signal.

Following the definition of each logical scheme as a sheet, for the sequential automata (a whole set of logical schemes) the metaphor of a "book" will be used (the concept of a book is already in use in modern spreadsheet software technology). Thus, the sheet represents each logical scheme, and the book (a set of sheets) represents the sequential automaton. Then one can refer to the functioning of the automaton as the process of "turning the book's pages".

To implement the "pages turning" process we suggest a special mechanism or macro-algorithm. The key function of this algorithm is the sequential calculation of the next corresponding sheet. The number of the next state (page or sheet number) is produced by the current sheet as it's output signal.

Table 1 is the FSM table representation of our example, the car-between-walls. It in fact comprises the set of subtables for the transitions among states. The book for the implementation of the *car-between-walls* device is shown in the Table 3. We

Table 3
Transition Table and "The Spreadsheet Book" for the Car-Between-Walls Device

ai	X(ai,aj)	aj	Y(ai,aj)
a1	x3'x1'x2'	a1	y0
	x3'x1'x2	a2	y2
	x3'x1	a3	y1
	x3	a1	y0
a2	x3'x1	a1	y1,y4
	x3'x1'	a2	y0
	x3	a1	y0
a3	x3'x2	a1	y2,y3
	x3'x2'	a3	y0
	x3	a1	y0

X1	X2	X3	Y1	Y2	Y3	Y4	a1	a2	a3
0	0	0	*	*	*	*	1	*	*
0	1	0	*	1	*	*	*	1	*
1	#	0	1	*	*	*	*	*	1
#	#	1	*	*	*	*	1	*	*

a1 a2 a3

X1	X2	X3	Y1	Y2	Y3	Y4	a1	a2	a3
1	#	0	1	*	*	1	1	*	*
0	#	0	*	*	*	*	*	1	*
#	#	1	*	*	*	*	*	*	1

a1 a2 a3

X1	X2	X3	Y1	Y2	Y3	Y4	a1	a2	a3
#	1	0	*	1	1	*	1	*	*
#	0	0	*	*	*	*	*	*	1
#	#	1	*	*	*	*	1	*	*

a1 a2 a3

want to emphasize that the bookmarks of the book indicate each state of the set of states of the automaton. The current state is shown in white color. This visual configuration supports a more clear understanding of the principles of control, and the tracing of the transition process among states.

5. FINAL REMARKS

Our main objective in the project reported in this paper is the gradual building of a substantial body of knowledge about both teaching and learning processes of control concepts and skills, fostering a successful integration of robotics into the curriculum. Our approach is that for achieving the goal, we should combine theoretical elaboration, research, curriculum development, and classroom implementation efforts. According to this approach we have elaborated the rationale or cognitive-curricular framework presented in this paper, we developed the Excel implementation of the different constructs, and we started a series of pilot studies based on teaching in a real-school-setting.

This work is aimed at having both short term as well as long term educational implications. In the short term we propose the incorporation of the conceptual approach (namely the different paradigms and control constructs) and of the computer environment (namely the Excel-based environment) into the learning activities. The conceptual model, as already described, offers teachers a characterization of different approaches to conceive represent and implement control, together with ways to translate these approaches into practical terms, definitions, notations, control-definition methods and sets of concepts to be taught. The computer environment provides students with a concrete tool for implementing control and for working with real models and devices. These two components may serve as tools for the development by teachers of their own set of learning activities, according to curricular needs and the students state of knowledge. For example, for our pilot implementation of the model we have developed a series of project-oriented instructional units. The unit -or project- comprises the building of a complex Lego device, and the planning and implementation of its control by the different paradigms or constructs. Experimental versions for two such units were just finished, one about an elevator system and another about an ambulatory robot-arm which can be designed to control a wide range of tasks (e.g., one project states the goal of cleaning a road and depositing all garbage in a given location). For the control part of the project, the students deal with the definition, notation and methodology regarding either the algorithmic or the state-diagram approaches, implement the control using the Excel-based interface, and run and debug it until the device's functioning proceeds according to the desired goals.

The long term educational implications of our project are related to an appropriate integration of robotic and control concepts in the school curriculum. Now-a-days these topics and concepts are no longer confined to professional programs (e.g., at high school specialization programs or higher education studies) but form part of the Israeli curriculum for all age levels. This reality implies that a whole set of solutions should be devised according to the needs and goals of the different

target populations, including conceptual approaches, teaching methods and learning materials and environments. Accordingly we are currently engaged in a long term research and development effort focusing on three age levels: kindergarten and early years of the elementary school, junior-high school, and high-school and college levels (which the project reported in this paper is a part of). The results at each stage take the form of both concrete products (e.g., control interfaces for young students or learning activities for working with autonomous robots) and a growing body of conclusions and knowledge about teaching and learning with these products.

Our hope and goal is that the knowledge and experience resulting from these combined efforts will contribute to the appropriate integration of robotics and control into the regular curriculum, and will serve as a model for approaching the teaching of new high-technological subjects at varied age levels in the future.

REFERENCES

[1] V. L. Parsegian, *This cybernetic world of men machines and earth systems*, Doubleday & Co., Garden City NY, 1972.

[2] H.A. Simon, The sciences of the artificial, MIT Press, Cambridge MA, 1981.

[3] D. Chen and W. Stroup, "Learning system thinking", *Tufts University - H. Dudley Wright Center*, Cambridge MA, 1992.

[4] J. R. Johnson, "Technology - report of the project 2061 phase I technology panel", American Association for the Advancement of Science, 1989.

[5] M. Resnick and S. Ocko, "LEGO/LOGO: Learning through and about design". In Constructionism, Edited by I. Harel and S. Papert, Ablex Publishing Corporation, Norwood NJ, 1991.

[6] N. Granott, "Puzzled minds and weird creatures: Phases in the spontaneous process of knowledge construction", In Constructionism, Edited by I. Harel and S. Papert, Ablex Publishing Corporation, Norwood NJ, 1991.

[7] E. Ackerman, "The agency model of transactions: Towards an understanding of children's theory of control". In Psychologie Génetique et Sciences Cognitives, Edited by J. Montangero and A. Tryphon, Foundation Archives Jean Piaget, Geneva, 1991.

[8] D. Mioduser, R. L. Venezky and B. Gong, "Student models of simple control systems" Computers in Human Behavior, in press.

[9] I. Levin and D. Mioduser, "A multiple constructs framework for teaching control concepts" IEEE Transactions on Education, in press.

[10] A. Falcione and H. Krogh, "Design recovery for relay ladder logic" IEEE Control Systems, April 1993, pp. 90-98.

[11] C. Fiedler and C. Alford, "Easy state tables for compound statement sequence detectors" IEEE Transactions on Education, Vol. 35. No. 3, 1992, pp. 243-246.

[12] I. Levin, "Matrix model of logical simulator within spreadsheet." *International Journal of Electrical Engineering*, Vol. 30, No. 3, 1993, pp. 216-223.