Computer Science Education
1998, Vol. 8, No. 3, pp. 181–196

0899-3408/98/0803–0181$12.00

Controlware for Learning Using Mobile Robots

Ilya Levin
School of Education, Tel-Aviv University

Vadim E. Levit
Center for Technological Education, Holon

ABSTRACT

One of the main problems faced within departments of computer science is how to encourage students to use their algorithmic and programming skills in the real-life context. This paper deals with control technologies for higher education. We define *controlware* as a toolkit for creating the control part of a mobile robot in interactive learning environments and educational micro-worlds. Intelligent behavior of mobile robots is created through the composition of constituent control units. The overall behavior of a mobile robot is half cooperative and half-competitive realization of the goals of the control units. To construct this composition behavior and resolve the corresponding conflicts, a partially ordering priority arbitration subsystem is defined. It is shown that the suggested approach links educational processes both in a control curriculum and an intelligent robotics curriculum. We hope that the spreadsheet-based controlware will make the mobile robots learning environment an accepted educational practice tool in computer science.

INTRODUCTION

The advent of mobile robots in the world of education has completely changed the standard approach to computer science instruction. For example, LEGO-LOGO systems are successful outgrowths of Papert's ideas on developmental education (Papert, 1980).

The success of the pedagogical process depends on a suitable agreement between the subject matter, technological means of teaching, and the formal models which are used during the teaching. In the present paper, a spreadsheet software environment and educational mobile robots are presented as the technological means, transition formulae as the formal model, and the

Correspondence: Dr. Ilya Levin, School of Education, Tel-Aviv University, Ramat Aviv, Tel Aviv 69978, Israel. E-mail: ilia1@post.tau.ac.il

digital control concept as subject matter. LEGO educational mobile robots are used to create a design-oriented learning environment (Jones & Flynn, 1993). We define mobile robots as computer-controlled devices that can move and interact with the external world.

We think that the goal-oriented behavior of mobile robots has to be created through the composition of some sort of components, which we consider as a generalization of electronic bricks. The overall behavior of a mobile robot is a combination of cooperative and competitive realizations of the goals of the constituent component bricks. To conduct this composition behavior and resolve the correspondent conflicts, a partially ordered set (*poset*) priority arbitration subsystem is defined.

We specify the concept of *controlware* as a combination of formal models of individual control units and a formal model of their composition. According to our approach in this paper, the formal model of a control unit is a transition formula; and the formal model of control units composition is partially ordered layered architecture.

As a high-level intelligent control paradigm within controlware we choose an architecture with five components: a set of independent constituent control units, a set of control instructions, a set of micro-operations, a partial order on this set, and a priority arbitration subsystem that chooses at every moment of time a subset of required micro-operations in accordance with the partial order and current decisions of the control units. A *poset priority arbitration subsystem* is regarded as a pair including a partial order and a choice function on the set of micro-operations determined by the control units serving as components of the mobile robot. It reflects a system of priorities imposed on the behavior of the mobile robot. The priority arbitration subsystem chooses a subset of decisions to be embodied in accordance with local decisions provided by the partial order. In this paper we restrict ourselves to priority arbitration subsystems with a partially ordered structure. This corresponds to a new approach to knowledge representation of the control instruction hierarchy as a generalization of standard priority systems. We consider this kind of architecture as a crucial part of controlware.

The assumption that the set of micro-operations of a mobile robot is a partially ordered set provides us with a polynomial complexity procedure for the realization of the priority arbitration subsystem. Every constituent control unit can be represented as a decision table. This kind of control system representation can be implemented as a spreadsheet-based matrix structure. LEGO mobile robots are used for hardware implementation of the operation unit of a controlled system.

This work is intended as an attempt to motivate a new flexible interface between goal-oriented behavior designers and mobile robots (Resnick, 1993). Our viewpoint is based on the discrete-event approach in teaching (Lewis, 1994), layered architecture (Brooks, 1986), and the spreadsheet-based method of implementation of the control automaton by decision tables (Levin, 1993).

Many systems devoted to educational or manufacturing activities are arranged into hierarchical structures, easily translated into partial ordering representation.

CONTROLWARE IN CONTROL CURRICULA

In this paper we discuss the present status of controlware and the general implications of our approach. In general, controlware differs from traditional software in the following significant ways:

1. While traditional programs manipulate data flows, controlware employs control flows.
2. Controlware exhibits a high degree of transparency; for instance, one can see immediately the role and importance of every subbehavior.
3. Controlware maintains a very flexible approach to mobile robots construction, which conforms with their incremental changes maintenance.
4. Controlware offers both a rich set of learning activities and a direct correspondence between the definition of mobile robots behaviors and their implementation environment.
5. While traditional software development is usually based on an algorithmic approach, controlware is based on the state–transition technique.
6. While traditional software is usually oriented on standard sequential computer architecture, controlware is based on parallel architecture.
7. A software program consists of a set of instructions, where each instruction causes the computer to carry out a certain operation. A controlware procedure comprises a set of intelligent bricks combined with a partial ordering arbitration priority system to resolve admissible combinations of contradicted intentions of the bricks.

What are the advantages and disadvantages of the controlware paradigm? By constructing independent intelligent bricks and combining them together under constraints in the form of a partial ordered priority system, the students develop a new kind of creative activity. They learn to reveal the different subcomponent

stages from the general design process and represent them in terms of an ensemble of constraints forming a partial ordered set of priorities. These two kinds of design correspond to the two known approaches to programming: top-down and bottom-up designs. Using partial orders as a simple and natural way of combining submodules and subprograms into a main program, the students are introduced to a new and fruitful flexibility in complex systems design. No longer do they see programming as a dull sequence of boring drills to be implemented in the frameworks of standard high-level programming languages. With controlware at hand the students immediately start from the new paradigm where programming of the control of intelligent vehicles does not resemble classical conservative programming in any way. Our approach is devoted to reconciling design and programming, creativity and painstaking recording of details, individualism and synergy.

A conservative programmer cannot foresee all the possible ramifications of difficulties in the process of program design. He normally faces an enormous number of debugging findings in the usual life cycle of his programs. Our student, armed with a kit of flexible controlware tools, can change, maintain, and check the structure of a mobile robot easily and safely, if needed.

The great advantage of controlware is its ability to produce and maintain a complex intelligent behavior as a straightforward consequence of simple and natural basic priority constraints.

Although the topics we have so far dealt with in this paper may appear to be somewhat disconnected, they are actually related to each other in their common concern with the relationship between synergy and individual behaviors and its influence on a Control System Curriculum.

Until recently, the traditional design of control units of complex systems and intelligent vehicles, in general, and mobile robots, in particular, has not contained both a state–transition and partial ordering approach to programming. These paradigms lead to a clear correspondence between the very high-level mental prototyping and the hardware level of implementation of complex intelligent goal-oriented behaviors. As a result, controlware will inspire students to advance the new educational relationships between programmable logic controllers, discrete-event control concepts, and the state-transition methodology on the one hand, and such pure mathematical ideas as boolean algebra, partial ordered sets, etc., on the other hand. These topics in their turn are of great use in the area of proper computer science, where they are treated as basic tools for the description and design of efficient algorithms. The teacher concerned with Control Curriculum organization and presentation

problems is often confronted with the choice of sequence for basic subjects. Taking into account that our approach to the implementation of controlware is based on a spreadsheet open environment, we propose to incorporate the following topics into Control System Curriculum: boolean algebra, theory of partial ordered sets, real-time programming, state–transition technique, discrete events control, homogeneous structures, and intelligent control systems.

Two of the courses should be specially mentioned. The courses 'Computer Systems in Education' and 'Development of Learning Environments' are being taught in the framework of technological teacher training as comprehensive courses, and in the framework of the Computer Science B.Sc.Ed. program at the Holon Center for Technological Education as elective ones. These courses are being given to a group of some twenty students, four hours per week during one year (one semester for each one). The lectures take place in a specially equipped computer laboratory. The students are normally divided into small groups (about three students per group). The data gathered from our teaching experience have become a background for research for two Ph.D. students and two M.Sc. students investigating the fields of Learning in Complex Environments and Educational Robotics.

FORMAL MODEL OF THE CONTROL UNIT

A broad spectrum of intelligent systems can be represented as compositions of control and operation units (Baranov, 1994). The set $X = x_1,x_2,\ldots,x_m$ of binary input signals is transferred from the operation unit to the control unit. The set of binary signals $Y = y_1,y_2,\ldots,y_n$ is the set of control instructions, transferred from the control unit of the system to the operation unit. The goal of the control unit is generation of the set of control instructions Y. The operation unit performs micro-operations in one-to-one correspondence with the set Y. We will not distinguish further between micro-operations and control instructions.

We define the formal model of the control unit not only as a mathematical model but also as a formal language for description of a family of rules of transformation, combination and minimization problems for a set of control units. From our point of view the existence of such a formal language opens great perspectives in the use of controlware in a classroom. In fact, formal transformations of control systems, if suggested to students as learning activities, are able to create an academic atmosphere similar to a mathematics class.

We introduce a special language of *transition formulae* as a formal model within controlware. Each control unit is associated with its corresponding transition formula. This transition formula is constructed as follows. Each boolean function α_i, depending on a set of variables $X = x_1,x_2,\ldots,x_m$, is put into correspondence with a micro-operation y_i. Function α_i is assumed to be equal to 1 if and only if micro-operation y_i is performed. Transition formula F associated with the whole control unit is defined as:

$$F = \sum_{i=1}^{n} \alpha_i y_i$$

where $\alpha_i y_i = y_i$ if $\alpha_i = 1$, and $\alpha_i y_i = 0$ if $\alpha_i = 0$.

It is important to check that the set of all α-functions is complete and orthogonal; that is:

$$\sum_{i=1}^{n} \alpha_i = 1$$

and $\alpha_i \,\&\, \alpha_i = 0$ $(k \neq j)$.

To represent simultaneous functioning of two different control units described by formulae F and G we define a product operation on the set of all transition formulae. Let

$$F = \sum_{i=1}^{n} \alpha_i y_i,\ G = \sum_{i=1}^{n} \beta_i y_i$$

The product of F and G is as follows:

$$H = F \times G = \sum_{i,j} (\alpha_i \,\&\, \beta_j)\ \{y_i, y_j\}$$

This formula exploits the usual approach to the idea of priority based on the supposition that the pairwise priorities are organized as a linear order on the set of micro-operations Y, which means that for every pair of micro-operations $\{y_i,y_j\}$ there exist only two options, shown as follows:

$\{y_i,y_j\} = y_i$, if $y_j > y_i$; and $\{y_i,y_j\} = y_j$, if $y_i < y_j$.

Assume that the three following formulae describe a control system:

$$F_1 = x_1y_1 + \bar{x}_1\bar{x}_2y_2 + \bar{x}_1x_2y_3,\ F_2 = x_1y_2 + \bar{x}_1y_3,\ F_3 = x_2y_3 + \bar{x}_2y_2$$

where $\bar{x}$ is used instead of function *not*(x), and the following linear order on the set of micro-operations is $y_1 > y_2 > y_3$. The product of formulae F_1, F_2, and F_3 describes the mutual functioning of their corresponding control units:

$$F = F_1 \times F_2 \times F_3 = (x_1y_1 + \bar{x}_1\bar{x}_2y_2 + \bar{x}_1x_2y_3) \times (x_1y_2 + \bar{x}_1y_3) \times (x_2y_3 + \bar{x}_2y_2) =$$
$$(x_1\{y_1,y_2\} + \bar{x}_1\bar{x}_2\{y_2,y_3\} + \bar{x}_1x_2y_3) \times (x_2y_3 + \bar{x}_2y_2) = x_1y_1 + \bar{x}_1\bar{x}_2y_2 + \bar{x}_1x_2y_3.$$

Each transition formula can be presented in table form. These tables are known as *decision tables* (Humby, 1973). Columns appearing in decision tables are marked by inputs $x_1,x_2, \ldots, x_m$ and micro-operations $y_1,y_2, \ldots, y_n$. Each term t_k of the transition formula is put into accordance with its corresponding row of the decision table. Character 1 appears at the intersection of row k and column x_h, if the variable x_h is contained in the term. Character 0 appears at the intersection of row k and column x_h, if $\bar{x}_h$ is contained in the term. Character "–" appears at the intersection of row k and column x_h if the variable x_h and its negative $\bar{x}_h$ are absent in the term. Character 1 appears at the point of the intersection of row k and column y_i if micro-operation y_i is contained in the term, and character "•" in the opposite case.

A control system defined by a set of transition formulae can be implemented using a *two level structure* of decision tables. The first level of this structure is a decision table implementation of the initial formulae, and the second level of the structure is a $Y \times (X \oplus Y)$ *priority decision table* (Kohavi, 1986), which means that every row of this table corresponds to an output from the set Y, the first part of the columns correspond to the set of inputs X, and the second part of the columns correspond to the set of outputs Y. For the control system containing the formulae F_1, F_2, and F_3 with the resulting transition formula F, and its corresponding priority decision table, this structure is illustrated by Figures 1 to 5 and Figure 7.

It is well known that the mathematical model of a control system is a finite state machine (FSM). Using the machinery of transition formulae as a mathematical model of a control system we describe only non-memory systems. So we have narrowed the class of systems which can be represented by the transition formulae model. The three following reasons led us to a decision to use the transition formulae as the basic model of control systems:

x_1	x_2	y_1	y_2	y_3
1	–	1	•	•
0	0	•	1	•
0	1	•	•	1

Fig. 1. The decision table of formula:
$F = x_1y_1 + \bar{x}_1\bar{x}_2y_2 + \bar{x}_1x_2y_3$.

x_1	x_2	y_1	y_2	y_3
1	–	1	•	•
0	1	•	•	1
0	0	•	1	•

Fig. 2. The decision table of formula:
$F_1 = x_1y_1 + \bar{x}_1\bar{x}_2y_2 + \bar{x}_1x_2y_3$.

x_1	x_2	y_1	y_2	y_3
0	–	•	•	1
1	–	•	1	•

Fig. 3. The decision table of formula:
$F_2 = x_1y_2 + \bar{x}_1y_3$.

x_1	x_2	y_1	y_2	y_3
–	0	•	1	•
–	1	•	•	1

Fig. 4. The decision table of formula:
$F_3 = x_2y_3 + \bar{x}_2y_2$.

Input values			Output values		
y_1	y_2	y_3	y_1	y_2	y_3
1	–	–	1	•	•
0	1	–	•	1	•
0	0	1	•	•	1

Fig 5. The priority decision table of the control system.

1. The class of systems which can be described by the transition formulae is wide enough to organize a rich variety of learning activities in a classroom of educational robotics.
2. The use of such straightforward notation as transition formulae reduces dramatically the amount of prerequisites required for the educational robotics course. The only knowledge needed with such an approach is a combination of some basic facts from boolean algebra and switching theory.
3. Moreover, if internal variables are accepted as external ones, the controlware, based on the transition formulae, can also be used for describing systems with memory. In this case special external memory devices must be included in the operating unit of the system.

Thus, we came to the conclusion that the transition formulae can be used instead of the classical FSM model without any loss of generality of the formal model of a control system.

FORMAL MODEL OF CONTROL UNITS COMPOSITION

Layered architecture comprises a family of basic independent components, known as layers. Each of them is capable of creating a subsystem behavior. Since decisions carried out by different layers may be in conflict with each other, the layered architecture should include a priority arbitration subsystem. The layered architecture (Brooks, 1986) is devoted to controlling mobile robots. This kind of architecture is intended to control a robot that wanders around the office areas of a laboratory building a map of its surroundings. The system is built up from layers to allow the robot to operate at increasing levels of competence. The system must specify exactly what must be done under all possible conditions. The layers are made up of asynchronous modules which communicate over low bandwidth channels. Each module is an exemplar of a fairly simple computational machine. Higher level layers can subsume the roles of lower levels by suppressing their output micro-operations. However, lower levels continue to function as higher levels are added. This approach results in robust and flexible control systems.

On the other hand, the disappearance of the internal variables has appeared to considerably simplify the model. Use of the transition formulae based controlware in the framework of Brooks' subsumption architecture instead of FSM resulted in elimination of the time variables from the classical model. In such a case it is unnecessary to include in the model the sophisticated concept of time-limitation of inhibition and suppression.

We can transform an informal approach to priorities into the rigorous notion of a partial order on the set of micro-operations. Let $Y = y_1,y_2,\ldots,y_n$ be the set of all micro-operations and the pair $\{Y, <\}$ be the partial order (*poset*) formalizing the natural idea of priority. We represent the $\{Y, <\}$ partial order by a $Y \times Y$ matrix A of the corresponding binary relation, where $A(i,j) = 1$ means that $y_i < y_j$ or $y_i = y_j$, $A(i,j) = 0$ means that $y_i > y_j$, and $A(i,j) =$ "–" means that y_i and y_j are incompatible.

Let us assume that a set of input values generates the set of micro-operations: $\hat{Y} \subseteq Y$. Every local y_i, y_j conflict can result in three ways: execute y_i and suppress y_j; execute y_j and suppress y_i; and execute both y_i and y_j in parallel. For each local conflict the arbiter chooses one of these alternatives. To dissolve the global conflict $\hat{Y}$ under the given family of priority constraints imposed by the partial order the arbiter has to choose only maximal elements of the set Y. Traversing the tree corresponding to the partial order $\{Y, <\}$ one can easily find all the maximal elements needed by an efficient polynomial algorithm.

Here we propose a decision table driven solution of the above problem in the framework of controlware. It is inspired by the $Y \times (X \oplus Y)$ representation of priority decision tables for transition formulae. This representation for describing a complex behavior has been presented in detail in the "Formal Model of the Control Unit" section. Now our intent is to show how to transform the $Y \times Y$ matrix of partial order on the set Y to the corresponding $Y \times (X \oplus Y)$ *poset decision table*, which means that every row of this table corresponds to an element of the set of outputs Y. The first part of the columns correspond to the same set Y interpreted as a set of inputs, and the second part of the columns corresponds to the set of outputs Y. To accomplish this task we assign to the first half of the $Y \times (X \oplus Y)$ poset decision table all the correspondent values of matrix A, changing them according to the following rules: "–" $\Rightarrow$ "–"; "0" $\Rightarrow$ "0"; not diagonal "1" $\Rightarrow$ "–"; diagonal "1" $\Rightarrow$ "1". The second half of the poset decision table is changed for the identity matrix. We will have more opportunities to represent this algorithm in a more quantitative way using an example of a mobile robot in the following section.

SPREADSHEET-LEGO ENVIRONMENT FOR IMPLEMENTATION OF THE CONTROL SYSTEM

We use one well-known type of modern software—spreadsheets—as a basis for decision tables implementation. Spreadsheets are a useful and flexible modeling tool. Spreadsheets may be regarded as a normal calculator-type tool, but they are also universal homogeneous two-dimensional fields which can be used for the implementation of various computational functions.

Levin's (1993) paper aims at studying the use of spreadsheets for simulating a particular type of integrated circuit—Programmable Logic Arrays (PLA). This simulation is especially interesting, since it allows the simulation of integrated circuits as well as anything lending itself to description by logical functions systems. The present study will also use the logical capacity of the spreadsheet as a base for building the systems of logical control.

The fragment of a spreadsheet which simulates the functioning of a PLA is the spreadsheet model of a certain decision table. Every cell (i, j) placed on the intersection of i-row and j-column of the spreadsheet model is programmed for the implementation of a logical function, according to the corresponding content of this decision table. This function is the universal function of the spreadsheet model. The representation of the PLA in the form of a spreadsheet model

is the simulation of the corresponding decision table. The spreadsheet simulations that are based on the matrix model are the matrix simulations. In fact, every transition formula can be represented as a decision table and implemented using the matrix simulation approach. Here we show that the matrix simulation approach is also suitable for implementation of compositions of control units; that is, for control systems design as a whole.

There is a direct connection between the spreadsheet method of decision tables implementation and parallel architecture solutions which rely on the ability to execute every decision rule independently, but this topic exceeds the scope of this paper.

A schematic description of the control unit in Figure 6 will serve as the basis for defining the matrix implementation of the transition formula for our example, as defined in the transition formula $F = x_1y_1 + \bar{x}_1\bar{x}_2y_2 + \bar{x}_1x_2y_3$ from the "Formal Model of the Control Unit" section.

Various implementations of control environments based on the homogeneous spreadsheet structure can be developed. One such environment—namely, the matrix two-level structure—has been chosen for our purposes. The schematic description of the matrix two-level structure of the example from the "Formal Model of the Control Unit" section is shown in Figure 7.

Another remarkable feature of the spreadsheet environment is its powerful graphical video-interface. It has been found that the spreadsheet approach is surprisingly suitable for implementation of control systems, while conventional software is merely an implementation thereof.

The learning environment was developed in EXCEL, taking advantage of its computational features for developing the required engines, and of its interface-design features (e.g., dialog boxes, buttons) for designing the working environment. EXCEL has been connected with the LEGO control interface (Control Lab) (Levin, 1994). Figure 8 shows the dialog boxes for entering input and output devices corresponding to x and y values.

In the input dialog box the student indicates, for each input (e.g., $x_1,x_2,\ldots,x_n$), the input port of the LEGO interface box to which the corresponding sensor is connected. In the output dialog box the student defines for each output (e.g., $y_1,y_2,\ldots,y_n$) the set of micro-operations, and the port to which specific activated elements (e.g., a motor, a lamp) are connected. In the framework of our EXCEL–LEGO environment the resulting spreadsheet becomes the control program of our mobile robot. To run the spreadsheet is to actuate the mechanism of behavior of the robot.

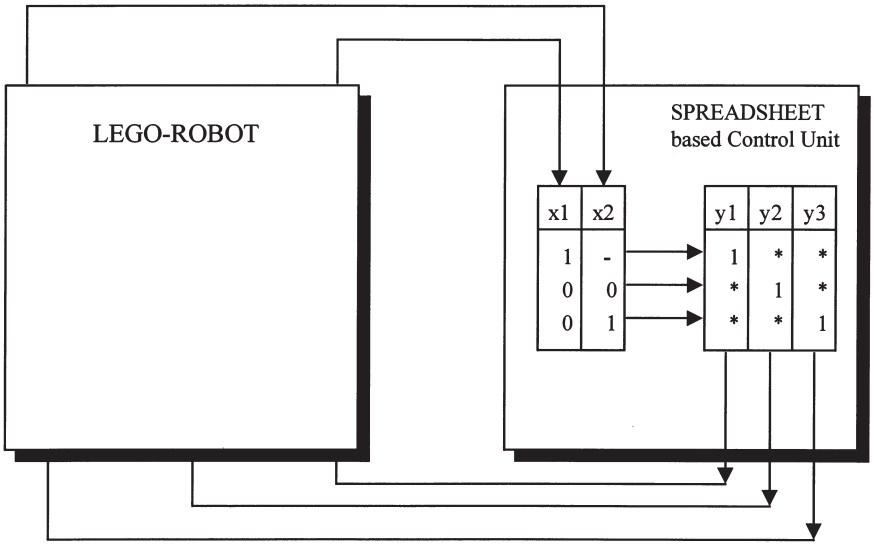

Fig. 6. Schematic description of the control unit.

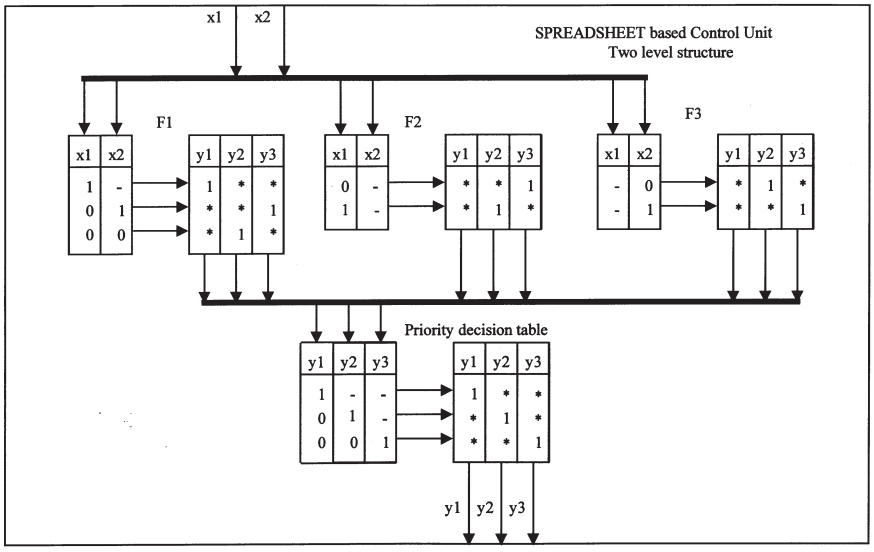

Fig. 7. Schematic description of the matrix two-level structure.

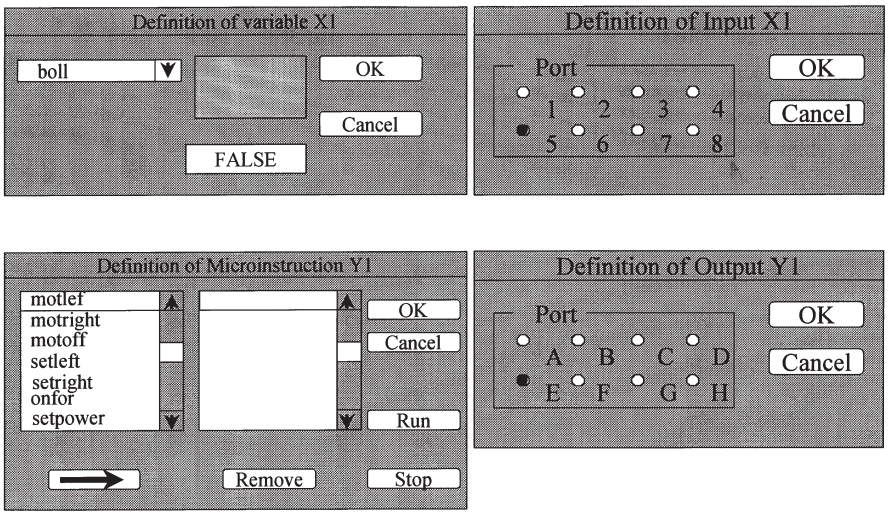

Fig. 8. Dialog boxes for input and output.

CASE STUDY

Here we will describe a mobile robot moving along the table in the direction of a light source. Assume that the robot includes only 5 sensors. Three of them are light sensors x_1, x_2 and x_3, positioned on the robot's front, left side and right side, respectively. Sensor x_4 detects the approach of the robot to the edge of the table. The fifth sensor x_5 detects the robot approaching the light source.

The robot is provided with a motor which can be controlled by the following micro-operations: y_1–to move the robot forward; y_2–to turn the robot to the left; y_3–to turn the robot to the right; y_4–to stop the robot; micro-operation y_5 shines a lamp on the robot, and y_0 is the empty micro-operation.

In accordance with the approach suggested in this paper, the following system of transition formulae may be built to describe the behavior of the robot:

$$F_1 = x_1y_1 + \bar{x}_1x_2y_2 + \bar{x}_1\bar{x}_2y_0$$

This means that if there is a light in front of the robot then it has to move forward. If there is no light in front of the robot, but there is a light to the left side of the robot, then it has to turn left. Otherwise the robot is supposed to suspend its activities (to perform the empty micro-operation).

$$F_2 = x_1y_1 + \bar{x}_1x_3y_3 + \bar{x}_1\bar{x}_3y_0$$

This means that if there is a light in front of the robot then it has to move forward. If there is no light in front of the robot, but there is a light to the right side of the robot, then it has to turn right. Otherwise the robot is supposed to suspend its activities (to perform the empty micro-operation).

$$F_3 = x_4y_4 + \bar{x}_4y_0$$

This means that if the robot has reached the edge of the table then the robot has to stop. Otherwise the robot is supposed to suspend its activities (to perform the empty micro-operation).

$$F_4 = x_5y_5 + \bar{x}_5y_0$$

	y_0	y_1	y_2	y_3	y_4	y_5
y_0	1	0	0	0	0	0
y_1	1	1	1	l	0	–
y_2	1	0	1	l	0	–
y_3	1	0	0	l	0	–
y_4	1	1	1	1	l	l
y_5	1	–	–	–	0	l

Fig. 9. The matrix of the partial order of the set of micro-operations Y.

x_1	x_2	y_1	y_2	y_3
1	–	1	•	•
0	1	•	l	•
0	0	•	•	•

Fig. 10. The decision table of formula: $F_1 = x_1y_1 + \bar{x}_1x_2y_2 + \bar{x}_1\bar{x}_2y_0$.

x_1	x_2	y_1	y_2	y_3
1	–	1	•	•
0	1	•	•	1
0	0	•	•	•

Fig. 11. The decision table of formula: $F_2 = x_1y_1 + \bar{x}_1x_3y_3 + \bar{x}_1\bar{x}_3y_0$.

x_4	y_4
1	1
0	•

Fig. 12. The decision table of formula: $F_3 = x_4y_4 + \bar{x}_4y_0$.

x_5	y_5
1	1
0	•

Fig. 13. The decision table of formula: $F_4 = x_5y_5 + \bar{x}_5y_0$.

Input values					Output values				
y_1	y_2	y_3	y_4	y_5	y_1	y_2	y_3	y_4	y_5
1	–	–	0	–	1	•	•	•	•
0	1	–	0	–	•	1	•	•	•
0	0	1	0	–	•	•	1	•	•
–	–	–	1	–	•	•	•	1	•
–	–	–	0	1	•	•	•	•	1

Fig. 14. The decision table of the poset: $y_0 < y_3 < y_2 < y_1 < y_4$, $y_0 < y_5 < y_4$.

This means that if the robot has started approaching the light source then it has to stop. Otherwise the robot is supposed to suspend its activities (to perform the empty micro-operation).

This example also includes the following partial order on the set of micro-operations:

$$Y = \{y_0,y_1,y_2,y_4,y_5\}: y_0 < y_3 < y_2 < y_1 < y_4,\ y_0 < y_5 < y_4.$$

This partial order is illustrated in Figure 9 by its corresponding binary relation matrix. The control part of the mobile robot is represented by the decision tables shown in Figures 10–13, and its poset decision table is shown in Figure 14.

The micro-operations which are the output values of the above decision tables shown in Figures 10–13 are the input values for the decision table corresponding to the partial order $\{Y, <\}$. The algorithm of transforming the matrix $Y \times Y$ of the partial order, shown in Figure 9, to its corresponding poset decision table $Y \times (X \oplus Y)$ (see Fig. 14) has been described in the section "Formal Model of Control Units Composition."

CONCLUSION

This paper has defined and examined the general properties of the controlware paradigm. In our approach a kind of trade-off is explored between the complexity of describing an intelligent behavior and the flexibility of a user-friendly interface. Our requirements of user-friendliness go beyond the standard requirements of the easy-to-adjust-to interface. Its real importance is displayed in the flexibility and creativity-inspiring properties of our poset–spreadsheet–LEGO environment. We believe that, for students, flexible creativity holds the key to the construction of complex intelligent vehicles. By incorporating flexible creativity possibilities into controlware systems one can alleviate not only computerized reality comprehension but also the student's progress in the world of discrete-events control.

It is important to point out that there are situations where the hypothesis of a partial ordered priority arbitration system is rather restrictive. The common problem for intellectual behavior modeling is determining a proper balance between the level of natural simplification and the capacity of the model.

No attempt has been made here to develop a model using priority systems as elementary building blocks. However, in future publications we intend to impose partial ordering restrictions only in localized areas.

A general model for describing complex behaviors of intelligent vehicles is presented here using the concept of controlware so that the priority arbitration subsystem can be analyzed and implemented. We have also investigated the properties of the partially ordered priority arbitration subsystem in detail and developed a spreadsheet implementation of our approach. Finally, we propose corresponding changes in the Control System Curriculum. We show that they lead to fruitful connections with both Computer Science Curriculum, in general, and LEGO techniques, in particular. Another important application of the controlware approach discussed here is the coordination and interconnection between Computer Science and Electrical Engineering Curricula via the Control System Curriculum.

REFERENCES

Baranov, S. (1994). *Logic synthesis for control automata*. Kluwer Academic Press.

Brooks, R. (1986). A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation, RA-2*, 14–23.

Humby, E. (1973). *Programs from decision tables*. London: Macdonald.

Jones, L.J., & Flynn, A.N. (1993). *Mobile robots: Inspiration to implementation*. Wellesly, MA: A.K. Peters.

Kohavi, Z. (1986). *Switching and finite automata theory*. Tata MacGraw-Hill.

Levin, I. (1993). Matrix model of logical simulator within spreadsheet. *International Journal of Electrical Engineering Education, 30*, 216–223.

Levin, I. (1994). The state machine paradigm and the spreadsheet learning environment. *Proceedings of Engineering Education Conference* (pp. 351–355). Sheffield Hallam University, Sheffield, England,.

Lewis, P. (1994). Introducing discrete-event control concept and state transaction methodology into control system curricula. *IEEE Transaction on Education, 37*, 65–70.

Papert, S. (1980). *Mindstorms: Children, computers and powerful ideas*. New York: Basic Books.

Resnick, M. (1993). Behavior construction kits. *Communications of ACM, 36* (7), 64–71.