The Determination of Results of Knowledge Test Control by Means of the Hierarchical Structures

Oleg V. Bisikalo, Robert G. Tadevosyan
Department of Economic Cybernetics and Informatics,
Vinnytsya State Agricultural University, Vynnytsya, Ukraine
E-mail: bisikalo@vsau.org, agoffice@svitonline.com

Abstract

Development of the methodology of designing of electronic textbooks with an opportunity of individual learning strategy's identification is considered in the article. In this regard a formal approach aimed at research of knowledge testing results is suggested on the basis of the hierarchical structures. Analytical decision of the objective has been elaborated and realised in practice in Common Lisp language.

Keywords: *electronic textbooks, testing, diversified lists, algebraic system, recursive algorithm, Common Lisp.*

Introduction

Providing the possibility of test results' combination as a concluding (synthetic) evaluation and analytical introduction of evaluation constituents aimed at the development of prospective educational strategy is the main objective of a test control of students' knowledge and skills [0]. The given task actually permits developing a dynamic model within the knowledge base of a test complex. The knowledge base can be regarded as a model of the subject matter area. The efficacy of utilizing electronic textbooks (ET) in the educational process primarily depends on the extent of educational material adaptation towards individual potential of a student. One can achieve a certain extent of electronic textbook adaptation through models and algorithms grounded on the theory of final automatons and the Markov chains. A model for providing management over adaptive training based on the theory of the Moore final automatons was used in the work [2] for the development of "Gefest" information-training system. A scenario "I won't let you go until teach you" is the core of the system. The essence of adaptation can be defined as the development of additional explanatory materials for students in case of shortcomings in basic knowledge and skills learned.

It should be mentioned that in the practice of the educational process, especially distance education, one could be faced with a problem when a student has already gained some skills and knowledge in the subject matter being taught. Time reduction on learning and motivation increase to work with electronic textbooks can be achieved for such students providing that a preliminary general test is conducted and individual training program is developed according to its results [0]. The mathematic model from [0] does not allow to do it though additional object model of the three-level development of the educational material, used in Gefest system, can partially facilitate to solve the problem.

Formalization of the subject matter and dynamic model of a student in the algebra aspect would allow incorporating the impact of substantial parameters of the distance education upon individual learning strategy and synthesizing appropriate adaptive algorithms of using ET under the conditions of real time.

The identification of analytical evaluation of a student test control aimed at the development of individual learning strategy is supposed to be our major objective.

Development of analytical evaluation algorithm

An electronic textbook can be represented as a combination of two constituents in the following formula:

$$Et = \langle Sa; Sd \rangle, \qquad (1)$$

where Et – electronic textbook model;
Sa – subject matter area model;
Sd – student model.

Let us assume that the structure of the electronic textbook is similar to the content-target structure of the educational material presented in [0].

It is believed that a typical electronic textbook consists of modules (sections) composed of topics, which in turn demonstrate a sequel of elementary educational doses. It is an example of the hierarchical structure as a model of the subject matter area in the form of a non-linear list:

$$\begin{aligned}&Sa=(1(11(111(1111)(1112))(112(1121)(1122)))\\&(12(121(1211)(1212)(1213))(122(1221))(123(1231)\\&(1232))(13(131(1311)(1312))(132(1321)(1322)))\end{aligned} \quad (2)$$

The list (2) can be demonstrated more visually in the form of the so-called structured print in Lisp programming language [0] (Addition 2).

In order to develop a formula of a student model Sd, that is to be created on the basis of the content-target structure test control of the educational material Sa, we will introduce the following algebraic system:

$$Sd=\langle Basis;Main\_Sign\rangle \quad (3)$$

where

$$Basis=\{ListStructures\} \quad (4)$$

end

$$Main\_Sign=\{IF,OP\}. \quad (5)$$

If the hierarchical structures are regarded as the basis of the algebraic system, then a signature Main_Sign as a combination of predicates and operations will be considered on two levels of abstraction, namely: high (macro-level) and low (micro-level). At first we will pay attention to a macro-level. Predicates' multiplication, determined by the sum of data being processed

$$\begin{aligned}IF=\{&BalanceBracket,\\&RightBracket,LeftAtomBracket\}\end{aligned} \quad (6)$$

and for operators' multiplication implemented data processing

$$\begin{aligned}OP=\{&CreateTop,AddMeaning,\\&EraseTop,Shift(QuantitySymbols)\}\end{aligned}, \quad (7)$$

where Predicates:

BalanceBracket – a true predicate if a number of open (left) brackets is equal to a number of closed (right) brackets in the list being analyzed;

RightBracket – a true predicate if the right bracket ")" is the next symbol in the list being analyzed;

LeftAtomRight – a true predicate in case when the left bracket – atom – the right bracket "(" Atom ")" are next symbols in the list being analyzed;

and Operators:

CreateTop – to create a stack top;

AddMeaning – to add the meaning to a stack top;

EraseTop – to erase a stack top;

Shift (QuantitySymbols) – to shift an indicator to the right according to a number of QuantitySymbols in the list being analyzed.

Let us examine now the two-based algebra

$$Algebra=\langle\{IF,OP\};Sign\rangle, \quad (8)$$

where there are two bases IF i OP which determine an operation signature Sign, the latter has been included logic (Boolean) operations, namely disjunction, conjunction, and negation, identified on IF base, as well as operations of composition, alternative and cycle, grounded on OP multiplication. The formula is as follows

$$Sign=\langle\{Or,And,Not\};\{*,([u]A,B),\{[u]A\}\}\rangle. \quad (9)$$

where

{ Or, And, Not } – main Boolean operations;

* – composition A * B is a consecutive implementation of operators: firstly - A, then - B;

([u] A, B) – alternative: in case of [u] operator A implements, in other case – operator B does;

{[u] A} – cycle: operator A will conduct iterative operation under u=0 until [u] becomes a true predicate.

It should be stressed upon, that operators

$$A,B\in OP \quad (10)$$

and a predicate

$$u\in IF \quad (11)$$

are meant for the above-mentioned Sign signature. Sign signature includes classic programming structures, which have been supplemented by Dijkstra E. W. in the technology of structured programming.

Algorithm idea on micro-level

A list is analyzed with the assistance of Shift () operator from the left to the right from the beginning to the end, i.e. until the moment when a number of open (left) brackets becomes equal to a number of closed (right) brackets. Algorithm's operation based on the use of the two-element n-address stack. As the recursive analysis goes, subordinate sublists are submitted to a stack in order to evaluate them on the basis of appropriate atoms and/or sublists included. The recursive evaluation is finished with corresponding extraction of a stack element at the time when only atoms are left to be subordinate on the given hub.

Algorithm's operation based on the use of the classic two-element n-address stack is shown in figure 1. The first element of the stack is used for atoms' preservation, the second one for recursive identification of their evaluation on the basis of sublists included and/or subordinate atoms.

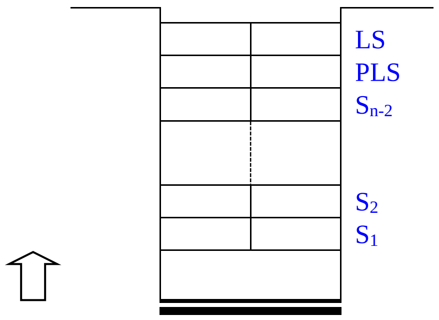

Figure 1. View of two-element n-address stack.

A recursive evaluation algorithm of the subject matter tree having seven terminal tops can be expressed by the structured scheme

$$\begin{aligned} &EVAL\_LIST ::= \{[\,BalanceBracket\,] \\ &([\,RightBracket\,]\,EraseTop, \\ &([\,LeftAtomRight\,]\,AddMeaning, CreateTop)) \\ &*\,Shift(\,QuantitySymbols\,)\} \end{aligned} . \quad (12)$$

$$\text{If } S = Su \cup So \quad (13)$$

is regarded as a system of algebra constituents, where

$$Su \subset IF, So \subset OP \quad (14)$$

then EVAL_LIST algorithm represents itself a composed operator in the form of term – superposition of elements with S and operations with Sign. The recursive algorithm can be graphically demonstrated the operation tree in figure 1.

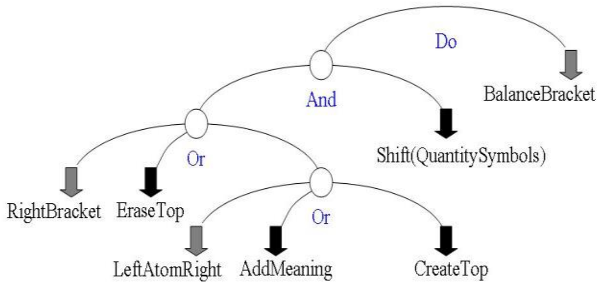

Figure 2. Recursive evaluation algorithm.

Macro-operators (7) can be in turn represented on the micro-level as elementary operators of processing hierarchical list structures

$$\begin{aligned} &CreateTop ::= \{\,Push * Ev(\,Atom\,) \rightarrow \\ &LS * Atom \rightarrow ALS * QuantitySymbols() = \\ &Atom + 1 * BalanceBracket() = \\ &BalanceBracket() + 1\,\} \end{aligned} , \quad (15)$$

where

Push – moving a stack down;

Ev(Atom) – atom evaluation identification (educational dose) according to test control results;

$\rightarrow$ – operator for meaning transfer while working with stack fields;

LS – field with a stack top meaning;

ALS – field with a stack top address;

QuantitySymbols () – function for identifying a number of symbols while shifting;

BalanceBracket () – function for identifying balance brackets;

$$\begin{aligned} &AddMeaning ::= \{\,Ev(\,Atom\,): \\ &LS = LS + Ev(\,Atom\,)* \\ &QuantitySymbols() = Atom + 2\,\} \end{aligned} ; \quad (16)$$

$$\begin{aligned} &EraseTop ::= \{\,ALS \rightarrow Atom * Ev(\,Atom\,): \\ &LS = LS + EV(\,Atom\,) * PLS = PLS + \\ &LS * Pop * QuantitySymbols() = 1 * \\ &BalanceBracket() = BalanceBracket() - 1\,\} \end{aligned} \quad (17)$$

where Pop – moving a stack up.

Analytical solution of the assignment obtained in the view of EVAL_LIST algorithm permits to synthesize individual learning strategies by means of incorporating the obtained assessment for each hub of the hierarchical ListStructures in the framework of the introduced Sd formal algebraic system.

Conclusion

General analytical decision of the objective concerning the determination of test control constituents (a student model development) according to the hierarchical structure of the educational material (a model of the subject matter area) has been elaborated in the work. It lays down the foundations of a synthesis of algorithm for developing individual learning strategy.

Algorithm's practical realization in Common Lisp programming language is demonstrated in Addition 1. The result of algorithm's work is shown in Addition 2.

References

[1] F. T. Tschang and T. Della Senta (eds.): Access to Knowledge: New Information Technologies and the Emergence of the Virtual University. Amsterdam: Elsevier Science and International Association of Universities, 2001, 167-206.

[2] Sh. Minasov, S.V. Tarkhov. Gefest Project as a Variant of Practical Implementation of E-Learning Technology at a Higher Educational Institution within Integration of Traditional and Distance Education / Educational Technology & Society, 8(1) 2005, p. 134-147 (in Russian).

[3] O.V. Bisikalo. Approach to the Development of Electronic Textbooks with Test Constituents Based on a

Model of Adaptive Education. "Control and Management in Complicated Systems" (CMCS - 2003) Collected Articles. Materials of VII International Research Conference. Vinnytsia, October 8-11, 2003. – Vinnytsia: UNIVERSUM – Vinnytsia, 2003. Pp. 227-234 (in Ukrainian).

[4] E. Khiuvenen, Y. Seppianen. The World of Lisp. 2 volumes. Volume I: Introduction to the Lisp Language and Functional Programming. Translated from the Finnish language. – M.: Mir, 1990. – 447p (in Russian).

Addition 1

```
(defun dft (tree n)
   (cond ((null tree) ())
         ((not (consp tree)) (format t "~a~%" (pad n
tree)))
         (t (dft (car tree) (1+ n))
            (dft (cdr tree) n))))
(defun pad (n s)
  (format nil "~a~a" (make-string n :initial-element
#\Space) s))
(defparameter *tt* '(1 (11 (111 (1111) (1112)) (112
(1121) (1122))) (12 (121 (1211) (1212) (1213)) (122
(1221)) (123 (1231) (1232))) (13 (131 (1311) (1312))
(132 (1321) (1322)))))
```

Addition 2

```
CL-USER> (dft *tt* 0)
 1
  11
   111
    1111
    1112
   112
    1121
    1122
  12
   121
    1211
    1212
    1213
   122
    1221
   123
    1231
    1232
  13
   131
    1311
    1312
   132
    1321
    1322
NIL
CL-USER>
```