Automation of Variant Preparation and Solving Estimation of Algorithmic Tasks for Virtual Laboratories Based on Automata Model

Mikhail S. Chezhin, Eugene A. Efimchik, and Andrey V. Lyamin(✉)

ITMO University, Saint Petersburg, Russia
{msch, efimchick}@cde.ifmo.ru, lyamin@mail.ifmo.ru

Abstract. In the work a description of an automata model of standard algorithm for constructing a correct solution of algorithmic tests is given. The described model allows a formal determination of the variant complexity of algorithmic test and serves as a basis for determining the complexity functions, including the collision concept – the situation of uncertainty, when a choice must be made upon fulfilling the task between the alternatives with various priorities. The influence of collisions on the automata model and its inner structure is described. The model and complexity functions are applied for virtual laboratories upon designing the algorithms of the variant constructing with a predetermined complexity in real time and algorithms of the procedures of students' solution estimation with respect to collisions. The results of the work are applied to the development of virtual laboratories, which are used in the practical part of massive online course on graph theory.

Keywords: E-learning · Virtual laboratories · Remote laboratory control protocol

1 Introduction

The development of information technologies in education resulted in a wide distribution of electronic teaching instruments, virtual laboratories (VL) being one of them. VL are electron media making possible the creation and study of visual models of the real phenomena. This is rather an extended determination taking into account the fact the both real laboratory installations (the laboratory is called distant in this case) and various mathematic and imitation models may form the basis of the models. A large number of information systems differing by aims, application methods, and program structure comply with it. A special feature of all VL may be their orientation to the formation and checking of practical skills and experiences. To determine the application field of the models and methods given in this work we shall give a short classification of VL.

One of the features of VL may be the type of tests embedded in them: there are both algorithmic tests requiring the fulfillment of a rigid sequence of actions and logical methods of solving and the tests associated with creative activity requiring the accomplishment of intuitive jumps, recognizing the objects, and the use of heuristic solving methods [1].

G. Vincenti et al. (Eds.): eLEOT 2015, LNICST 160, pp. 35–43, 2016.
DOI: 10.1007/978-3-319-28883-3_5

Another feature may be the program architecture of VL. Autonomic VL are a united supplement [2, 3], and the functions of distributed VL are divided between several individual modules interacting between each other with the help of special Remote Laboratory Control Protocol (RLCP) [4] or other network technologies [5].

An important marker is the presence and the method of automatic checking student's solution, since this property directly influences the applicability of VL during independent work with electronic practical courses of electronic information and education media. The automatic checking is especially important for massive open online courses. Automatic estimation is often carried out with the help of a method of testing the black box: student's solution is represented as a system, which can be acted upon and its reaction can be compared with what was expected [1, 6, 7]. One more example of commonly used practice in the systems of massive online courses may be the instrument of peer assessment: after completing his own test the student must check several solutions of other course participants selected randomly [8]. The method of checking depends on the character of the test: the black box test is convenient and therefore widely used method of automatic checking of the tests concerning the description of a designated algorithm in some program or modeling language and a peer assessment is used when a student must present a work badly amenable to automatic analysis – an essay, figure, or abstract.

One more property of VL is its wide application. Multi-purpose VL [1, 3], which can be used to carry out the studies on various topics, are usually substantially more scaled and complex than specialized VL developed for solving the problems concerning one topic [2].

In [4] a model of a distributed VL of a standardized structure with automatic checking students' solutions, which represents VL as a connected up modulus of electronic information-education system, is given. This model allows creation of a unified medium of fulfillment for multiple VL on the basis of AcademicNT system. Such a method appeared to be appropriate for control over the software of small specialized VL intended mainly for the work with algorithmic tests. We emphasize that the student is not required to describe the very algorithm, but he must reproduce its actions correctly for a given variant. Preparation of the test variants in the automatic way appeared to be an important problem – it was decided to reject the traditional bank of variants owing to its inherent drawbacks. In this case, variants of tests must have a predetermined complexity to ensure equal conditions for all students in the estimation of achieved education results. The complexity of the test variant in this context is a quantitative characteristic reflecting the number of operations needed to be fulfilled for obtaining a correct solution. It is necessary to distinguish the complexity of the test variant from the difficulty of the test. The difficulty is associated with mastering the algorithm of the test solving and is expressed by a percent from the number of students being tested from a representative selection, who fulfilled this test correctly. Nevertheless, under condition of a limited time the complexity of the test influences its difficulty. Even knowing the algorithm of solving the test you can fail to meet the schedule of its fulfillment, if a given variant has an excessive complexity.

2 Automata Model of Reference Algorithm

In this work a method is proposed for formal determination of the complexity of variants for algorithmic tests based on automation model of *reference algorithm*. Let us assume that there is a certain algorithmic test t, which must be solved with the help of reference algorithm a, and there is a great number of variants V, for it:

$$V = \{v_1, v_2, v_3, \cdots, v_n\}. \tag{1}$$

Each element v_i is a particular variant of the test with specific data.

As an example let us concern ourselves with an algorithm for Turing machine, which increases an integer by a unit. The starting number is in the tape in the binary digits written from left to right, in other cells this is an empty symbol, and the head points to the eldest order of the number. Then with the aim of increasing the number by a unit we must fulfill the following sequence of actions: (1) Move to the right till you meet the empty symbol; (2) Shift to the left; (3) If symbol in the current cell equals '1', change it for '0' and move to the left; (4) If the value of the current cell equals '0' or an empty symbol, write down '1' into the cell and complete the work. In this case the test t requires the actions of the reference algorithm a to be reproduced, and the starting state of Turing tape is a variant of test v_i. Here we should emphasize once more that the student must not write the program for Turing machine but must reproduce the above described algorithm correctly. He gains access to Turing tape, the possibility to accomplish the requests for reading a symbol from a current cell and the commands for shifting the head and writing the symbol into a cell.

To develop VL with automated processes of constructing the test variants and estimation of the student solutions for a model test t with the help of algorithm a we suggest to advance a special automation model M. As such model we propose to use a combination of determined final automate with an output (controlling automate) and a data depositary, which it interacts (the control object) with. This model is the development of a model of automated object (AO).

The AO model consists of three main components: controlling automate, object of control and external medium. At every step of the work the controlling automate forms a new state of the control object (calculating state) on the basis of external medium action, of current state of the control object and of the state of controlling automate (controlling state). Applying this model to algorithmic tests we find that at every step the controlling automate forms a record of a correct solution s, on the basis of the data of the test variant v, of the intermediate results of previous steps fixed in the record of the solution. The test variant is the object of the external medium, and the state of the control object must be considered as the record of the test solution including intermediate results. This means that after completion of the work of the controlling automate the object of control must contain all the information about the transfers carried out by the controlling automate and about the sequence of control states attended by it. Then with the help of the advanced model M we can obtain a correct solution for each variant of the test v. In other words, there is a reflection ρ_M of a great number of the test variants V to a multitude of solutions S (ρ_M: $V \rightarrow S$).

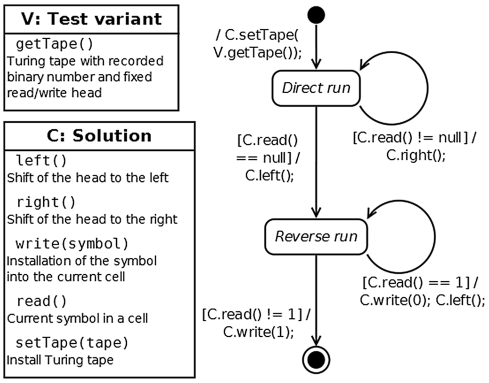

Fig. 1. Automation model of the algorithm of increasing an integer by a unit for Turing machine

Figure 1 represents the AO for the algorithm of increasing an integer by a unit described above. The solution being formed is contained in the control object *C*. The variant of the test *V* contains the starting state of Turing tape, which can be obtained with the help of the getTape inquiry.

A special attention must be paid to the structure of the object of control, since it must be designed in such a manner that in the resulting calculating state all the intermediate results were given. Hence, the object contains Turing tape, which starting state can be found with the help of the setTape command. The control over the tape is performed with the help of the head shift commands (left and right), the record of symbols (write) and an inquiry for reading the symbol in a current cell (read). Moreover, in order to save the intermediate results a journal of accomplished commands is added into the control object. Each time a shift of the head of the symbol record is performed, a corresponding record is added to the journal. Such a journal may be presented as one more Turing tape, let us call it the tape of command journal, with the aim of distinguishing it from the data tape. As the automate fulfills the command to shift the head of the data tape or to record a symbol, a symbol is written into the current cell of the command journal, which designate this command, then the head of the tape of the command journal is shifted to the right. The structure of the control object is given in Fig. 2. Table 1 contains description of commands and inquiries of both test

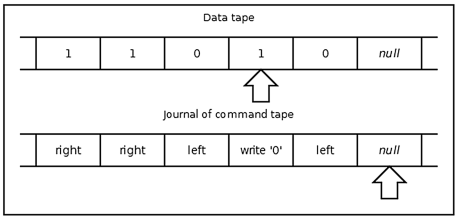

Fig. 2. Structure of the control object of automation model of the algorithm of increasing an integer by a unit for Turing machine Data tape, journal of command tape

Table 1. Commands of the test variant and the solution being formed

Object	Command	Description
V: Test variant	getTape()	Turing tape with recorded binary number and fixed head
C: Solution	left()	Shift of the head to the left
	right()	Shift of the head to the right
	write (symbol)	Installation of the symbol into the current cell
	read()	Current symbol in a cell
	setTape (tape)	To install Turing tape

variant and solution being formed in control object. The model is loaded into the controlling automate, the operation of copying the state of Turing tape from the test variant into the solution being formed is included into the initiating stage in this case.

Assuming that a student must adhere to reference algorithm *a* and its representing with the control object initiated according to the data of a variant and also an interface for interacting with it. We can reason that in the case, when the student reproduced the actions of the algorithm correctly, his solution as a resulting calculating state of the given control object must coincide with the correct solution. Thus, after adding a unit to "11011" the data tape must contain "11100" and the tape of the journal of commands – the sequence "right, right, right, right, right, left, write '0', left, write '0', left, write '1'".

The aforesaid interface must be designed in such a manner that the student could interact with the control object with the help of the same commands as the controlling automate. The student has an access to the same commands, which are used in the automation model in Fig. 1 – the head shift and symbol recording. Moreover, the student only sees the content of the current cell, as well as the controlling automate, hence the solution of the variant is not evident for him, and he must follow the given algorithm.

3 Consideration of Collisions

The special feature of automation models under consideration is the following: external actions on AO are known to be determined first of all by the data of the test variant. Moreover, some algorithms as known allow a selection from several equivalent alternatives (collision) during fulfillment, and then great number of correct solutions may exist for one variant of the test. In this case student's solution may be correct, but not coinciding with the solution constructed with the help of automation model. To avoid such situation it is necessary that the automation model take into account student's choice made in the situation of collision. A correct solution is constructed with the help of the controlling automate as long as the situation of collision emerges. Then the information about the made choice is selected from student's solution and is added to the input action on the automate at the next step of the work. Then the record of the solution is constructed by the usual manner up to the emergence of the next collision. Therefore, we can state that each component of the input action x_E on the controlling automate really consists of the component x_{EV} resulting from the test variant and the

component x_{EU} resulting from the student's choice fixed in the solution suggested by him: $X_E \subseteq X_{EV} \times X_{EU}$. Therefore in the general case the reflection ρ_M of great number of test variants to a multitude of the solutions given as M must take into account student's choice in the situations of collision and is determined as function $\rho_M : V \times U \rightarrow S$, where U is the multitude of the choices made by the student in the situations of collision.

In the example considered above the collisions are not encountered, however they can appear, if the test were a little changed and, for example, let the student from the beginning set the head into any number order but the eldest. This means that the model of the control object given in Fig. 2 must be updated by adding one more element – a number variables into which the information must be written about the very number order the student set the head into. In this case a modification of the automate model of the algorithm given in Fig. 1 must be made: one more element appears in the model – student's solution U. The model U and the model of the solution C coincide, however they have various sets of commands. It is possible to find out into which number order the student set the head with the help of getInitPosition() inquiry of the student solution U. A command for setting the head getInitPosition() is added to the solution C, it is excited at the stage of initiation. The modified automate model is given in Fig. 3.

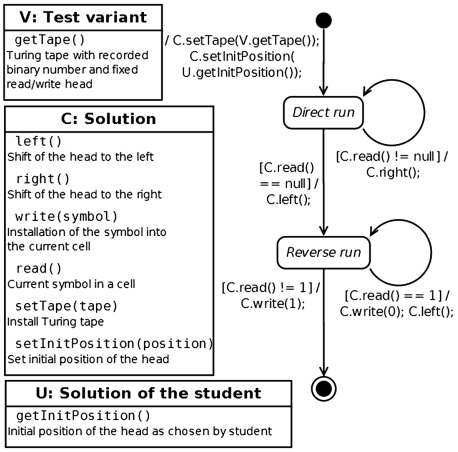

Fig. 3. Automation model of the algorithm of increasing an integer by a unit for Turing machine taking account of collisions

4 Formal Determining of Complexity of Algorithmic Tests

The complexity of the test variant c_v can be determined as the number of transfers completed by the controlling automate if the complexity of completing the transfers is the same. This value can be obtained as the number of terms q of the sequence Y_v of controlling states visited by the controlling automate in the process of constructing the correct solution for the test variant v:

$$\begin{aligned} Y_v &= (y_{r_i})_{i=1}^{q}, \\ y_{r_i} &= \delta\left(y_{r_{(i-1)}}, x_i\right), \\ c_v &= q, \end{aligned} \tag{2}$$

where δ: $X \times Y \rightarrow Y$ is the transfers function of controlling automate of model M; x_i – an input action on the controlling automate formed under the influence of variant v in the i – cycle; r_i – the index of the visited controlling state.

In the case when the complexity of completing the transfers cannot be considered the same, it is necessary to determine the function f of complexity of completing the transfer c_i to the state y_{r_i} from the state $y_{r_{i-1}}$ under the action x_i. Then the resulting complexity of the test variant will be equal to the sum of the complexities of the transfers made by the controlling automate:

$$\begin{aligned} Y_v &= (y_{r_i})_{i=1}^{q}, \\ y_{r_i} &= \delta(y_{r_{i-1}}, x_i), \\ c_i &= f(y_{r_{i-1}}, y_{r_i}, x_i), \\ c_v &= \sum_{i=1}^{q} c_i. \end{aligned} \tag{3}$$

Using such a procedure we can determine the complexity of any existing variant of the test, but this cannot solve the problem of automatic construction of the test variants of a designated complexity. It is necessary to examine the automation model of algorithm for each test t in order to answer the question, which properties of the test variants do influence the complexity of the solution and how they influence. The result of such examination must be the function of the kind $c = c(v)$, making possible the calculation of the complexity as a function of the properties of the test variant. Let us call it *complexity function*.

In the example described above (without admission of the collisions) the number of transfers completed by the controlling automate depends on two factors: l – the total number of the number orders and z – the ordinal number of the last number order equal to '0'. Automation model at the stage of direct run passes all the number orders and at the stage of reverse run completes one more transfer in order to return to the last order and passes to the first order not equal to '1', hence the resulting complexity is: $c = l + 1 + (l - z) = 2l + 1 - z$. For example, if a number «1000000000» is written in Turing tape, the complexity of adding a unit to it will be $c = 2 \times 10 + 1 - 10 = 11$. Really, the stage of reverse run includes only one transfer. For the variant with number «101111» the complexity is also equal to 11: $c = 2 \times 6 + 1–2 = 11$, since in the stage of reverse run there will be five transfers.

5 Automatic Variant Constructing and Estimation of the Student Solution

Having the complexity function, we can compose the algorithm of constructing the variants of algorithmic test with a designated complexity. Thus, if it is necessary to create the variant of the test for reproducing the above described algorithm of the increment of binary number with a designated complexity C, we can use the algorithm given in Fig. 4.

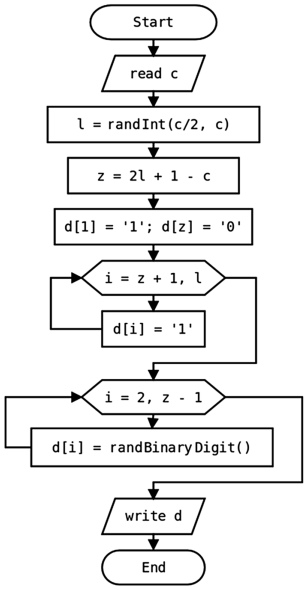

Fig. 4. Algorithm of constructing the test variants

6 Conclusion

To summarize we can point out that the automation model of reference algorithm allows the complexity of the variants of algorithmic tests to be determined formally. Nevertheless, to solve the problem of automatic construction of the test variants with designated complexity it is necessary to determine the complexity function, which characterizes the dependence of the variant complexity on its properties. Automation of composing the variants with equal complexity an of estimating on the basis of verification procedure allows VL to be created for algorithmic tests functioning in the completely automatic regime, which makes possible their use in the preparation of massive online courses.

Automation model of reference algorithm allows us to correlate the student's solutions with a fortiori correct solution making possible the checking of all the

intermediate results and a precise answer to the question whether the student presented a correct solution and also an indication the place of an error. This model gives us the method of formal determination of the variant complexity of the algorithmic test. The significance of this method consists in the fact that on its basis the algorithms of constructing the test variants with designated complexity are created. However, the field of application of this model is limited by algorithmic tests only.

References

1. Lyamin, A.V., Vashenkov, O.E.: Virtual environment and instruments for student olympiad on cybernetics. In: Proceedings of 8th IFAC Symposium on Advances in Control Education, pp. 95–100, Kumamoto, Japan (2009)
2. Tao, J., Jing-ying, Z., Lang, W.: The thermal simulation of electromechanical platform system. In: Transportation Electrification Asia-Pacific (ITEC Asia-Pacific), 2014 IEEE Conference and Expo, pp. 1–4, Beijing (2014)
3. Jaffry, D.: Best Practices for Implementing Modeling Guidelines in Simulink, http://www.mathworks.com/company/newsletters/articles/best-practices-for-implementing-modeling-guidelines-in-simulink.html. Mathworks (2014)
4. Efimchik, E.A., Lyamin, A.V.: RLCP-compatible virtual laboratories. In: The International Conference on E-Learning and E-Technologies in Education (ICEEE 2012) Proceedings, pp. 59–64, Lodz, Poland (2012)
5. Xu, L., Huang, D., Tsai, W.-T.: Cloud-based virtual laboratory for network security education. IEEE Trans. Edu. **57**(3), 145–150 (2014). IEEE Press, New York
6. Fu, Q., He, K., Ma, X.: Research on experimental skills assessment based on computer simulation technology. J. China Distance Educ. (3), 68–71 (2005). Beijing
7. Rodríguez-del-Pino, J.C., Rubio-Royo, E., Hernández-Figueroa, Z.J.: A virtual programming lab for moodle with automatic assessment and anti-plagiarism features. In: Proceedings of the 2012 International Conference on e-Learning, e-Business, Enterprise Information Systems, & e-Government, Las Vegas (2012)
8. Sterbini, A., Temperini, M.: Peer-assessment and grading of open answers in a web-based e-learning setting. In: Information Technology Based Higher Education and Training (ITHET), pp. 1–7, Antalya, Turkey (2013)