Visual Tools to Support the Study of Abstract Calculators

Tamara M. Kuzmina[a] and Olga A. Vetrova[b]
A. N. Kosygin RSPU (Technologies. Design. Art), 117997, Sadovnicheskaya street, 33, building 1, Moscow, Russian Federation
{kuzmina-tm, vetrova-oa}@rguk.ru

Keywords: Visual tools, Support of the study, Abstract calculators, Turing machine, Deterministic finite automaton, Knowledge check, Computer software.

Abstract: The study of abstract calculators or abstract computing devices (Turing machines, normal Markov algorithms, finite automata) is included in a number of disciplines for training future IT professionals. The article discusses application programs that visualize the operation of these devices. Such programs greatly simplify the study of abstract calculators. In addition, using the considered programs as simulators, students not only get acquainted with the principles of operation of a Turing machine or a finite automaton, but also with a completely unusual way of programming for them.

1 INTRODUCTION

The integration of new information technologies in education has led to the emergence of new educational technologies and forms of education based on electronic means of processing and transmitting information. The appearance of powerful computer multimedia systems became the basis for the creation of interactive computer programs, visualizing educational material, and caused the intensive development of modern education. At the junction of pedagogy and programming, a new direction of scientific activity has emerged.

The idea of visualization in pedagogy is not new. Throughout its history, humanity has used it to transfer knowledge to the younger generation. Also, in ancient times, teachers noted that the more feelings are involved in the transfer of knowledge, the more successful this transfer is. In the XVII century, the educator Ya.A. Komensky, systematizing the experience accumulated in the field of education, argued: "... let it be a golden rule for teachers: everything that can be provided for perception by the senses. If any objects can be perceived by several senses at once, let them be immediately grasped by several senses" [Komensky, 1982, P. 384]. Later, in the XIX century, K.D. Ushinsky, studying the factors affecting the quality of learning and the development of logical thinking, came to the same opinion: "The more our sense organs take part in the perception of an impression, the more firmly these impressions fall into our mechanical, nervous memory and are easier to remember later" [Ushinsky, 1950, P. 251]. Visualization involves vision and thereby increases the power of influence on the learner.

Visualization is understood as a way of obtaining and generalizing knowledge based on a visual image of a concept, event, phenomenon, fact, algorithm, etc., based on associative thinking and systematic structuring of information in a visual form. Moreover, visualization of algorithms is also understood as their animation. In the modern world, visualization is usually understood as computer visualization, which begins its history with the 1987 report [Visualization in Scientific Computing, 1987], and is developing very rapidly to this day. Visualization is considered both as a modern computer method of analyzing various scientific data when conducting theoretical and experimental research [Isaev, Podvesovskii, 2022; Klyshinsky, Rysakov, Shikhov, 2014; Pilyugin, Malikova, Pas'ko, Adzhiev, 2012; Korneev, Shalyto, 2005], and as a strategically important direction for the development of education [Tikhomirov, Sal'dikov, Malikova, Kuchenkova, Pilyugin, 2012; Kuzmina, Vetrova, 2022; Danilov, 2022]. It should be noted that these approaches

[a] https://orcid.org/0000-0001-5872-8107
[b] https://orcid.org/0000-0001-6935-0787

overlap in many ways [Zakharova, Shklyar, 2013], their difference primarily lies in the fact that with the latter approach we are talking about the visualization of already systematized studied knowledge.

The visualization process is the folding of various types of information into a visual image, in addition, the process of this folding is ambiguous. There may be different visual images of the same source. Visualization is directly related to modeling and, therefore, to the consideration of different sides of the studied object. In pedagogy, this process depends not only on the material under consideration, but also on the audience for which the visualization is being created, its level of preparedness and interest.

The proposed article discusses application programs that demonstrate the work of abstract calculators, such as the Turing machine [Kuzmina, Doronina, 2008], normal Markov algorithms [Kuzmina, 2023] and finite automata [Kuzmina, 2010]. For today, there are a large number of similar programs. In each case, these programs reflect not only the work of the computer itself, but also the vision of this work by the authors, because abstract devices are conceptual (theoretical). In our opinion, the presented emulators reflect the work of calculators [Polyakov, 2023; Alistat, 2023; Github, 2023] too formally without a game moment allowed in teaching. The formation of an abstract concept is a rather complex multi-stage process: the stage of perception (when information is acquired at the level of senses), the stage of representation (when some features of the object being studied are realized). You can speed up this process with the help of visual images.

2 APPROACHES TO VISUALIZE THE WORK OF ABSTRACT CALCULATORS

2.1 Visualization of the Operation of the Turing Machine

A Turing machine is an abstract computing device that was invented by Alan Turing to formalize the concept of an algorithm. According to the Church-Turing thesis, if the problem is algorithmically solvable, it is possible to build a Turing machine that solves this problem. And, conversely, for problems for which it is impossible to build a Turing machine, there is no solution algorithm.

The concept of a Turing machine is abstract and difficult for students to perceive, and an application program, in which a difficult-to-perceive model is presented in the form of some device performing specified actions, helps to understand everything. This program allows you to enter unusual commands, that define the actions of the model, and see, how these commands are executed.

When studying the discipline "Mathematical Logic and Theory of Algorithms", students are required not only to consider working models, but also to be able to develop a set of commands for a Turing machine solving a particular problem. According to the Church-Turing thesis, all problems solved with the help of a computer can be solved by building a Turing machine. But programming when building a Turing machine is very different from programming using classical languages, at least in that the program for the Turing machine does not take into account the order of commands at all.

Perhaps that is why the teacher, when checking the program that sets the operation of the Turing machine, faces an unexpected problem: when finding an error, sometimes quite obvious, it turns out to be very difficult to explain its presence to the student. To do this, you need to give an example and demonstrate on it how the program will be wrong. There are a lot of students, their mistakes are diverse, and the student is interested primarily in why his job is not accepted, and not the work of a neighbor. Therefore, teachers often limit themselves to very simple tasks, not because students will not be able to cope with more complex ones, but out of fear of getting stuck with justifying grades. This problem is solved by using such programs, since during the animation the student perfectly sees, how the device is wrong and why it does not work as we would like.

The Turing machine is a conceptual device, no one can tell what it "really looks like". Since its "construction" took place in the first half of the last century, we presented it in the form of an ancient lamp apparatus. In the theory of algorithms, it is assumed that the tape of the machine is stationary, and the reading - writing head is moving. But since the movement is relative, we considered it possible to make changes to the design of the machine and consider the moving tape.

The reading-writing head is the upper part of the device, an infinite tape passes through it, divided into cells. Only one cell is always viewed, the analyzed symbol is displayed on the upper screen (Figure 1).

The lower screen displays the internal state of the machine and the executed command, if the machine is running. Before changing the internal state, its record changes color.

In theory, it is believed that a set of commands defines a Turing machine. Therefore, when our program is running, and when the command editor window is open, we see an extended panel with lamps, we can assume that this is how we change the device of the machine itself.

To record a word on the tape, this word is entered in the input field "Initial recording on the tape". This entry can only be changed when the machine is stopped. The external alphabet of the machine can consist of any set of Latin and Russian letters, as well as numbers. The digit "0" is used as the symbol of an empty cell. The program executes commands step by step, and the student can consider the moment when the error appears and think about how to revise it.

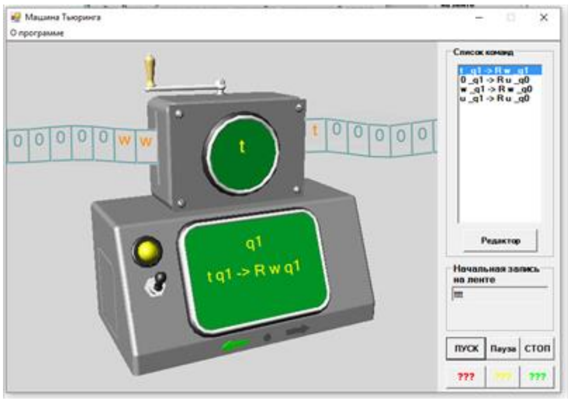

Figure 1: The Turing machine executes the command.

2.2 Emulator of Normal Markov Algorithms

Since normal Markov algorithms (NMA) are studied in the discipline "Mathematical Logic and Theory of Algorithms", an application program was written that demonstrates the work of NMA [Kuzmina, 2023].

An interesting feature for NMA is that they use only one elementary action – the so-called substitution.

A substitution formula is a record of the form $\alpha \rightarrow \beta$ (it is read as "α is replaced by β"), where α and β are any words (possibly empty). In this case, α is called the left part of the formula, and β is called the right part.

The substitution itself, as an action, is given by the substitution formula and is applied to some word *P*. The essence of the operation is that in the word *P* a part is found that coincides with the left part of this formula, and it is replaced by the right part of the formula. At the same time, the remaining parts of the word *P* do not change. The resulting word *R* is called the result of substitution. The substitution may be normal, or it may be final.

A normal Markov algorithm is a nonempty finite ordered set of substitution formulas called the NMA scheme. To write down the algorithm in the form of NMA means to draw up a scheme for NMA.

The operation of the normal Markov algorithm is reduced to the execution of a sequence of steps. At each step, the substitution formulas included in the normal Markov algorithm are viewed from top to bottom and the first of the formulas applicable to the input word *P* is selected. Next, the substitution is performed according to the found formula. It turns out a new word *P'*. If the substitution is final, then the procedure is terminated, and if the substitution is normal, then we proceed to the next step. At the next step, this word *P'* is taken as the input one and the same procedure is applied to it, a new word *P''* is obtained, and so on, before either the final substitution is not applied, or a situation arises where no substitution will fit the word being processed. In any case, the normal Markov algorithm will finish the work. The resulting word will be the answer.

The software "Control and training program on the topic "Normal Markov algorithms", as the name implies, can work in training and in control modes. In the control mode, a task is displayed to the student on the screen, the student enters the scheme to the normal Markov algorithm, and the program checks his work on testing examples. If the scheme has passed the test, then the work ends there. But if errors are found, then these errors can be viewed in more detail. The student chooses a word that his scheme converts incorrectly. First, all substitutions, performed according to the scheme proposed by the student, are output. If the student fully understands the mistake made, then the work can be terminated. If the student is interested in the details, he can view the calculations in step-by-step mode. If the work with some substitution turns out to be incomprehensible, this work can be viewed in a separate window, which shows an animation demonstrating the search for the possibility of using a substitution for a word, and the result of the application, if the possibility of substitution is found (Figure 2).

In the training mode, there is no control task, but instead, some sample is offered that can be considered at all levels of detail. You can enter your own example, but you will also have to enter your own solution.

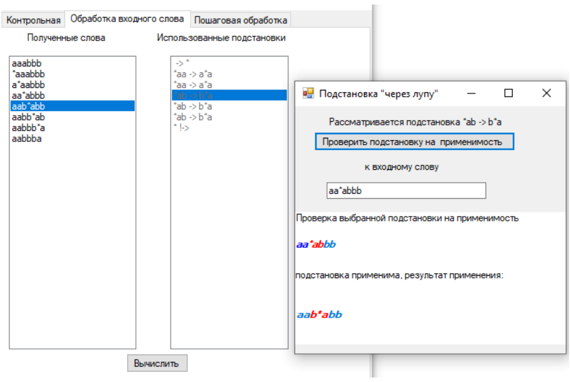

Figure 2: Emulator of NMA. Checking the applicability of a specific substitution.

2.3 Visualization of the Operation of a Deterministic Finite Automaton

Students study finite automata much later than Turing machines or normal Markov algorithms, in the discipline "Linguistic and information support of logistics systems", and therefore, they do not experience great difficulties in studying. But the tasks for students are more difficult, it is necessary not just to build a deterministic finite automaton (DFA), but to write a computer program that simulates the operation of this automaton. Errors can appear both at the stage of DFA development and at the stage of writing a computer program. Errors in the development of DFA can be found easier, but it is more troublesome to fix them if the code has already been written. Therefore, students are offered the computer application that visualizes the work of the DFA, which will allow them to find errors in the development of the DFA before writing the code, or after writing it, when receiving incorrect results during the tests of their program.

When launching the considered application program [Kuzmina, 2010], you first need to determine the alphabet, then the program itself will build a table with the column headers of which will be the symbols of this alphabet. The student needs to fill out a table defining the work of the DFA. Since errors are possible at this stage, the table is checked by the program for correctness. After the DFA transition table is accepted by the program, the student sets the initial and final states. In Figure 3 is an example of an automaton that accepts the words "star" and "start", this automaton has only one final state, and in general there may be several of them. The initial state of the DFA is only one.

After the automaton is defined, you can view its operation (Figure 3). Just as when demonstrating the operation of the Turing machine, the DFA tape was made moving, and the device was stationary. But the DFA looks different than the Turing machine. The reviewed word is divided into two parts, to the left of the reading head is the analyzed part, to the right is the unanalyzed part of the word. The symbol that is being analyzed is shifted closer to the analyzing device at this particular moment. The current state of the DFA is shown on the control device, the new state can be seen in the transition table, it is marked with a red frame. The execution takes place step by step, when you click the "One action DFA" button.

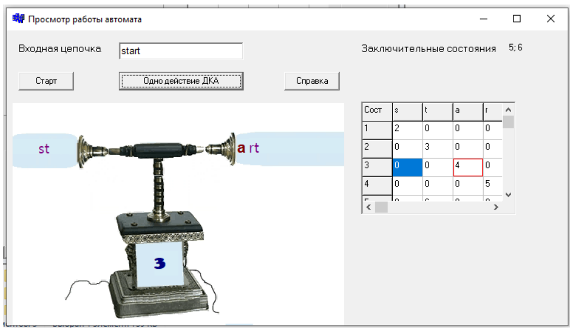

Figure 3: Visualization of DFA operation.

3 CONCLUSIONS

It should be noted that the use of the considered practical visualization programs made it possible to raise the bar for student work assessments. If works were accepted on paper in which inaccuracies and even certain errors were allowed, then when working with these visualizers, we require a program that works correctly. Students write and debug small programs of 10-20 lines to the state of full operability and use these examples to get acquainted not only with the principles of the Turing machine or NMA, but also with completely unusual programming methods, when writing programs that set the work of Turing machines, the order of commands does not matter, and when creating schemes, the order of substitutions is important to NMA, but is used not like in programs written in familiar programming languages.

The use of the program demonstrating the work of DFA has significantly reduced the time that students spend on laboratory work, and the answers to questions on DFA have become more detailed during the exam. In addition, all the considered visualizers free the teacher from searching for errors in the "paper" versions of student works. The considered software can be attributed to simulator programs (trainers).

REFERENCES

Alistat, 2023. *Online Turing Machine Simulator*, Https://alistat.eu/online/turingmachinesimulator.

Danilov, O.E., 2022. Solving the visualization problem to ensure visibility in the educational process. In *The collection: Modern trends in the development of science and the world community in the era of digitalization. Collection of materials of the VII International Scientific and Practical Conference, P. 178-185.* LIMITED LIABILITY COMPANY "IROK", IP OVCHINNIKOV MIKHAIL ARTUROVICH (ALEF PRINTING HOUSE).

Github, 2023. *Turing Machine Emulator*, Https://github.com/vnesp-donnu/turing.

Isaev, R.A., Podvesovskii, A.G., 2022. Cognitive Clarity of Graph Models: an Approach to Understanding the Idea and a Way to Identify Influencing Factors Based on Visual Analysis. In *Scientific Visualization, Vol. 14, No. 4, P. 38 – 51.* NATIONAL RESEARCH NUCLEAR UNIVERSITY "MEPHI", MOSCOW.

Klyshinsky, E.S., Rysakov, S.V., Shikhov, A.I., 2014. Review of methods of visualization of multidimensional data. In *New information technologies in automated systems, No. 17, P. 519-530.* NATIONAL RESEARCH UNIVERSITY HIGHER SCHOOL OF ECONOMICS, MOSCOW.

Komensky, Ya.A., 1982. *Selected pedagogical works. Volume 1. The Great Didactics,* Pedagogy. Moscow.

Korneev, G.A., Shalyto, A.A., 2005. Construction of visualizers of algorithms of discrete mathematics. In *Scientific and Technical Bulletin of the St. Petersburg State University of Information Technologies, Mechanics and Optics, No. 23, P. 118-129.* SAINT PETERSBURG NATIONAL RESEARCH OF INFORMATION TECHNOLOGIES, MECHANICS AND OPTICS.

Kuzmina, T.M., 2010. The program "Electronic textbook "Deterministic finite automaton". In *Certificate of official registration of the program for JeVM.* RU 2010615841.

Kuzmina, T.M., 2023. Monitoring and training program on the topic "Normal Markov algorithms". In *Certificate of registration of the program for JeVM.* RU 2023611108.

Kuzmina, T.M., Doronina K.A., 2008. The program "Electronic manual "Turing Machine". In *Certificate of official registration of the program for JeVM.* RU 2008611848.

Kuzmina, T.M., Vetrova, O.A., 2022. Graph visualization in the development of the knowledge testing program on graph theory. In *Scientific Visualization, Vol. 14, No. 1, P. 41-49*. NATIONAL RESEARCH NUCLEAR UNIVERSITY "MEPHI", MOSCOW.

Pilyugin, V., Malikova, E., Pas'ko, A., Adzhiev, V., 2012. Scientific visualization as a method of analyzing scientific data. In *Scientific Visualization, Vol. 4, No. 4, P. 56-70*. NATIONAL RESEARCH NUCLEAR UNIVERSITY "MEPHI", MOSCOW.

Polyakov, K., 2023. *Educational computer model "Turing Machine"*, Http://kpolyakov.spb.ru/prog/turing.htm.

Tikhomirov, G., Sal'dikov, I., Malikova, E., Kuchenkova, L., Pilyugin, V., 2012. The experience of NRU MEPHI in the development and use of visualization software in the educational process in the field of nuclear power plants. In *Scientific visualization, Vol. 4, No. 2, P. 57-63*. NATIONAL RESEARCH NUCLEAR UNIVERSITY "MEPHI", MOSCOW.

Ushinsky, K. D., 1950. *Collected works. Vol. 8: A human as a subject of education. The experience of pedagogical anthropology,* Academy of Pedagogical Sciences of the RSFSR, Institute of Theory and History of Pedagogy. Moscow-Leningrad.

Visualization in Scientific Computing, Special Issue, 1987. In *ACM SIGGRAPH Computer Graphics, Vol. 21, No. 6.* SPRINGER.

Zakharova, A., Shklyar, A., 2013. Metaphors of visualization. In *Scientific visualization, Vol. 5, No. 2, P. 16-24*. NATIONAL RESEARCH NUCLEAR UNIVERSITY "MEPHI", MOSCOW.