Automated verification of expression transformation chains based on computational experiments

Fedor Novikov¹, Viktor Katsman¹ and Vladislav Mosin²

¹ Peter the Great St.Petersburg Polytechnic University, Saint-Petersburg, Russia
² National Research University Higher School of Economics, Saint-Petersburg, Russia

E-mail: fedornovikov51@gmail.com, vikto9494@gmail.com, vmos1999@mail.ru

Abstract. The solution of mathematical and logical problems is the imperative element of the educational process in the technical sciences. To solidify the assimilated skills, learner need to solve a large number of educational tasks, that leads to the need of checking of a large number of solutions. In lots of cases the solution of an educational task is the chain of expression transformations. Many systems allow to compare the learner’s answer with the correct one, defined by the teacher. But in some cases, the teacher needs to set some additional information, besides the answer expression. Such additional information cannot be set for each possible pair of expressions in each possible solution chain of the task. We propose a method for verifying task solutions, based on the computational experiments between each pair of steps in solution chain. The proposed approach was tested and showed good results. Namely, in the experiments the efficiency of the teacher's work when checking solutions increased more than 4 times.

1. Introduction

Nowadays, automation is one of the most effective, and widespread ways to increase the efficiency of the learning process. In particular, automatic education environment systems become much more popular. Such systems provide the functionality to perform the part of the teacher's work or significantly simplify it, allowing teachers to concentrate on the most interesting and important aspects of their work. Many routine actions can be performed by computers, which will not get annoyed or tired, unlike living teachers. What we have to do, is just to think once, but very carefully, on what is to be done by a learner. After that machines will work and check if a learner is right or not. It gives us many advantages: machines work is much easier to scale, and the quality of machines work does not depend on their personal specificity and mood. Quality of verification depends only on the original algorithm embedded in a machine [1-3].

Different aspects of the teacher's work could be automated: discipline maintenance during classes, new material presentation, answers and solutions checking, and others. Automation is already actively used for many of these aspects, for example, during the lesson some part of the teacher's instruction is replaced by a recorded video. In online learning systems, video recording allows the teacher to replicate their instruction. Such automation requires the teacher only one-time careful preparation of the video, instead of repeating instruction time and time again.

It is much more difficult to automate the work of a teacher in situations where an individual approach to each learner is required, taking into account mood, level of knowledge and psychology. In
any case, after initial instructions and explanations, it is necessary to consolidate them, that is, it is necessary to make the learner to solve a large number of tasks.

It is desirable to select the tasks individually for each learner so that to solve the task, he or she has to apply the newly acquired skills, as well as their poorest skills from the recent training course. In addition to selecting tasks for learners, it is also necessary to check their response and solutions. It is eligible that when receiving the results of the test, the learner could understand where exactly he or she could have made a mistake, and, perhaps, could also get a clue how this error should be amended.

The opportunity to verify educational tasks solutions automatically, taking into account the requirements described in the previous paragraph, opens a lot of prospects in the field of teaching. In particular, it reduces the time that spends the teacher to check solutions. In addition to the diversity of full-time education, this opportunity also has applications in self-education and distance learning.

In this paper, we describe a method that allows checking educational task solutions automatically. The method is particularly well suited for many tasks in elementary algebra, trigonometry, physics, informatics, and many others. We validate the proposed method on the several groups of junior students of the Peter the Great Saint-Petersburg Polytechnic University on the tasks in areas of set theory and combinatorics.

2. Existing methods of exercises automation

To date, checking the answer is a common method of automated verification of the solution of the problem. In this case, the teacher manually sets the condition of the problem and the correct answer criteria, which will determine the correctness of the answer given by the learner. Accordingly, the task is counted for the learner if and only if his or her answer meets the criteria set by the teacher.

Criteria for the correctness of the answer might differ. The simplest criterion for the correct answer is the correct answer itself. In the field of natural or technical sciences, it is usually a number, math expression or some formatted sequence of symbols. If the answer can be written in different ways, for example, with different symbolic expressions or different precision, then you can either supplement the condition with the rules for formatting an answer or allow the learner to pick one of the multiple-choice options. Both approaches theoretically allow the teacher to test the answer in any task and are widely used in distance learning, for example, on the Coursera [4], EdX [5], Stepic [6], WebWork [7], Moodle [8] and many others.

But, in many cases, these methods significantly increase the routine component of the problem-solving process. For example, if the learner is asked to choose the right expression from the suggested ones, he or she will be required to read all the suggested expressions, including those which will not lead to the desired result. Also, the answer can be guessed by chance, and the skill will not be assimilated.

In some cases, even answers written in different ways can also be unambiguously compared. For example, the equality of two symbolic expressions could be checked automatically if these expressions contain only natural numbers and variables that are linked with addition, multiplication and subtraction. Tarski's algorithm allows us to verify the truth of closed arithmetic formulae of the first order with variables for real numbers, that is, with a finite set of real numbers [9].

But for most problems there are no simple ways to check: for example, the impossibility of automatically checking the equivalence of two programs by their code follows from Rice's theorem [10]. The impossibility of automatically verifying the equality of symbolic expressions using rational numbers and the operations of exponentiation follows from Richardson's theorem [11]. Thus, we cannot verify the correct answer in all cases using the precise methods only.

Probabilistic methods provide us more possibilities. Namely, in many cases, this problem of the ambiguity of the answer form might be solved by the testing method. The learner's answer compares with the correct one by performing a series of computational experiments, in each of which the result obtained by the learner is compared with the correct answer at certain values of the parameters: symbolic variables in the expression. If in all computational experiments the learner's answer
coincides with the correct one, the solution is considered to be correct, otherwise, you can give the learner a counterexample.

Such a verification method is probabilistic since an incorrect answer can be interpreted as correct because the test coverage cannot be complete. So, the selection of the values of the parameters for which it is required to perform computational experiments (test cases) becomes the key factor. The teacher needs either to conduct it manually - to work out the tests, or to provide an algorithm for this selection - to design the generation of the automatic test. Both methods require serious effort. Nevertheless, this method of verification is widely used in distance learning systems, especially for software testing. For example, when creating online courses in the Stepic system, you can ask the learners to write programs, and you can use automatic tests as a test criterion [12].

For example, when testing the statement “\(a - b = a / b\)”, the test “\(a = 4; b = 2\)” passes, and the test “\(a = b = 4\)” does not.

In addition to the chance of interpreting the wrong answer as correct, such testing has another drawback, namely the lack of form verification. However, the form of the answer is often important. When programming it is important to follow the style of coding. When specifying a symbolic expression in an answer, it is eligible to simplify it. Sometimes it is required to verify if the learner has or has not applied a certain operation in the solution.

In most school tasks the methods listed above make it possible to automatically verify the correctness of the answer, and sometimes to show the learner where he or she made a mistake. But all these methods allow us to check only the final result without analyzing how the learner has obtained it. In fact, it often happens that learners come to the right answer by chance, without taking into account a number of important facts and inferences. If you check not only the value of the answer but the solution itself, you can help the learner find the point from which his or her reasoning became erroneous.

In this article, we propose an approach significantly expands the prospects for automated verification of tasks solutions.

3. Proposed computational experiments method

Two expressions can be compared by performing a series of computational experiments, in each of which the values of the expressions are compared at randomly chosen values of the symbolic variables in the expressions. If in all computational experiments values of the expressions are the same, the expressions are considered to be equal, otherwise not.

Sometimes counterexample can be missed and unequal expressions can be interpreted as equal. The probability of such error significantly increases if the expressions have a small domain, so the values of the expressions at randomly chosen values of the symbolic variables can be undefined.

For instance, an expression “\(\ln(0-|x|)\)” is undefined at a randomly chosen value of variable “\(x\)” with 100% probability. Such expression would be considered as equal to any other expression with the empty domain, for example to “\(\ln(0-|x|) + 1\)”.

Calculation complex function value, which is always defined, is usually used to solve this problem. This method is based on Identity Theorem. Namely, two holomorphic functions, equal on a set with a limit point, are equal on a whole domain [13].

However, the method of checking expressions equality in complex numbers has a problem: complex logarithm is a multi-valued function with an infinite set of values [14]. Checking equality of two infinite sets requires too much computation resources [15]. For solving this problem, we propose to fix one holomorphic branch of logarithm \([0, 2\pi]\).

The proposed approach of expressions comparison is as follows:

- The system chooses sets of values of the symbolic variables in the expressions by the uniform distribution.
- For each set of values of the variables system checks equality of the expressions in real numbers.
If the system finds a counterexample, in which expressions are different, then expressions are considered as unequal. Otherwise and if expressions are defined in all sets of values, expressions are considered as equal.

For each set of values of variables in which expressions are undefined, the system checks equality of the expressions in complex numbers.

The system estimates the probability for expressions being equal and compares it with a ratio of the number of points where expressions are equal to the number of points where expressions differ. If the probability is bigger, expressions are considered as equal, otherwise unequal.

3.1. Probability for expressions being equal estimation.

Consider the equality:

\[ \ln \left( \prod_{j=1}^{n} z_j \right) = \sum_{j=1}^{n} \ln(z_j) \]

, where \( z_i \) is a complex number in exponential form, \( z_i = r_i e^{i \phi_i} \), \( r_i > 0 \). Then \( \ln(z_i) = \ln(r_i) + i \phi_i \).

Consider the logarithm of the production:

\[
\ln \left( \prod_{j=1}^{n} z_j \right) = \ln \left( e^{i \sum_{j=1}^{n} \phi_j} \prod_{j=1}^{n} r_j \right) = \ln \left( \prod_{j=1}^{n} r_j \right) + i \left( \sum_{j=1}^{n} \phi_j - 2\pi k \right) = \\
= \sum_{j=1}^{n} \ln(r_j) + i \left( \sum_{j=1}^{n} \phi_j - 2\pi k \right)
\]

, where \( k \in \mathbb{Z} \) is selected to satisfy \( 0 \leq \sum_{j=1}^{n} \phi_j - 2\pi k < 2\pi \), because one holomorphic branch of the logarithm \([0, 2\pi]\) is fixed.

Consider the sum of the logarithms:

\[ \sum_{j=1}^{n} \ln(z_j) = \sum_{j=1}^{n} \left( \ln(r_j) + i \phi_j \right) = \sum_{j=1}^{n} \ln(r_j) + i \sum_{j=1}^{n} \phi_j \]

The difference between the logarithm of the production and the sum of the logarithms happens if and only if \( k \neq 0 \), because of the fixation of the concrete logarithm branch. Consequently, the probability for expressions being equal is the probability of \( k = 0 \), that means \( 0 \leq \sum_{j=1}^{n} \phi_j < 2\pi \).

Let \( \xi_j \) be a random variable, the angle of j-th complex number. Let \( p_{\xi_j}(x) \) be the density of the distribution, \( p_{\xi_j} = \frac{1}{2\pi} \) because the distribution is uniform.

**Lemma.** For two logarithms (n=2) the probability is \( \frac{1}{2} \).

**Prove:**
\[ P(\xi_1 + \xi_2 \leq 2\pi) = \int_0^{2\pi} \int_0^{2\pi-x} p_{\xi_1}(x) p_{\xi_2}(x) \, dx \, dy = \int_0^{2\pi} \int_0^{2\pi-x} \frac{1}{4\pi^2} \, dx \, dy = \frac{1}{2} \]

**Lemma.** For \( n \) logarithms the probability is \( \frac{1}{n!} \).

The distribution of \( n \)-dimensional vector \((\xi_1, \xi_2, ..., \xi_n)\) is uniform distribution in a \( n \)-dimensional cube with edge \( 2\pi \).

\[ P(\xi_1 + \xi_2 + \cdots + \xi_n \leq 2\pi) = \text{ratio of the volume of the set } X = \{x = (x_1, x_2, ..., x_n) \mid x_i \geq 0, \sum_{i=1}^{n} x_i < 2\pi \} \text{ to volume of all } n \text{-dimensional cube.} \]

The volume of the cube is \( (2\pi)^n \).

Set \( X \) corresponds to the pyramid, which is cut from the cube by a hyperplane \( \sum_{i=1}^{n} x_i = 2\pi \). The hyperplane cuts off the \( n \)-dimensional pyramid with an edge \( 2\pi \) incidental to the straight angle. The volume of such \( n \)-dimensional pyramid is \( \frac{(2\pi)^n}{n!} \). The probability to get point from the pyramid is the volume of the pyramid divided by volume of the cube \( \frac{(2\pi)^n}{n!} \).

So, the probability for expressions being equal is \( \frac{1}{n!} \) where \( n \) corresponds to the number of logarithms in the expressions.

4. Experiments

The experiments were conducted to make sure that the proposed method is applicable not only for teachers but for learners as well. Namely, it is important to verify that:

- the results obtained by automated verification of solutions will be close to the results deduced by the teacher;
- the problem solving will take approximately the same time to put it into the computer as it takes to write the solution on paper.

So far, we have already managed to conduct experiments on 3 groups of 2nd-year students of the Department of Applied Mathematics of the Peter the Great Saint-Petersburg Polytechnic University within the framework of the course "Discrete Mathematics" in 2017, 2018 and 2019. During these years, the testing program has been significantly improved: logic was improved, the number of allowed ways of recording solutions was increased, some bugs were fixed, but the overall concept of experiments has not been changed. In the framework of experiments, firstly the students were briefly explained the principles of the system, and then they were claimed to solve several problems in combinatorics and set theory.

The results show that students can quickly absorb the new rules for solutions notation. The speed of problem solving on the computer on average turned out to be about one and a half times less than the speed of solving problems on paper. The speed of recording the solution was increased in the last year because of significant improvements in notation and interface. Most important that, the number of ways of the recording solutions was significantly increased, new possibilities allowed to omit a lot of unnecessary details in the recording.

Another goal of the experiment was to get an idea of how much time the teacher saves with the help of automated verification. It took about an hour to set the tasks (one-time routine), plus half-hour was spent on bringing together the results of the automated verification. Manual verification of solutions of the same tasks made by the reference group of students took more than 6 hours.
Thus, the conducted experiments confirm the possibility and relevance of using the proposed method for automated verification of solutions of standard educational problems.

5. Conclusion
The proposed extension for the computational experiments method significantly expands the prospects for verifying solutions of typical educational tasks. The results of using this method substantially depend on the degree of ease with which the learner perceives the recording of problem solutions, which should also be understood by the checking system. The proposed extension significantly increases the number of ways of the recording solutions, allows to omit unnecessary details.

We keep working in such areas, our future prospects of the research are to support new function and operation types in new subject areas and to continue to improve the notation for the recording of solutions.

References
[1] Auvinen T 2015 Harmful Study Habits in Online Learning Environments with Automatic Assessment. Proceedings of the 2015 International Conference on Learning and Teaching in Computing and Engineering pp 50-7
[2] Willman S, Linden R, Kaila E, Rajala T and Laakso M 2015 On study habits on an introductory course on programming Computer Science Education 25(3) 276-91
[3] Tirronen M and Tirronen V 2016 A framework for evaluating student interaction with automatically assessed exercises Koli Calling '16 Proceedings of the 16th Koli Calling International Conference on Computing Education Research pp 180-1
[4] 2019 Coursera Homepage Retrieved from: https://ru.coursera.org/
[5] 2019 EDX Homepage Retrieved from: https://www.edx.org/
[6] 2019 Stepik Homepage Retrieved from: https://welcome.stepik.org/en
[7] 2019 WeBWorK Homepage Retrieved from: https://webwork.clearning.ubc.ca/webwork2/
[8] 2019 Moodle Homepage Retrieved from: https://moodle.org/
[9] Rogers H 1987 The Theory of Recursive Functions and Effective Computability (MIT Press)
[10] Rice H 1953 Classes of Recursively Enumerable Sets and Their Decision Problems Transactions of the American Mathematical Society 74(2) 358
[11] Richardson D 1968 Some Unsolvable Problems Involving Elementary Functions of a Real Variable Journal of Symbolic Logic 33(4) 514-20
[12] 2019 Stepic Code Lesson Retrieved from: https://stepik.org/lesson/9173/step/1?course=Epic-Guide-to-Stepik&unit_id=1718
[13] Markushevich A I 2005 Theory of functions of a Complex Variable (New York: American Mathematical Society) pp 108-14
[14] Sarason D 2007 Complex Function Theory (American Mathematical Society)
[15] Mathews J and Walker R L 1970 Mathematical Methods of Physics (MA: W. A. Benjamin/Addison-Wesley) pp 450-71
[16] 2015 Analytic functions of one complex variable Encyclopedia of Mathematics (European Mathematical Society fl. Springer)
[17] Baiguskarov T, Khabibulin B and Khasanova A 2017 The logarithm of the modulus of a holomorphic function as a minorant for a subharmonic function Mathematical Notes 101(3-4) 590-607