A NEW COURSE 'ALGEBRA + COMPUTER SCIENCE': WHAT SHOULD BE ITS OUTCOMES AND WHERE IT SHOULD START

ALEXANDRE BOROVIK AND VLADIMIR KONDRATIEV

Abstract The words "Programming is the second literacy" were coined more than 40 years ago [13], but never came to life. The paper develops and details that old slogan by proposing that the mainstream mathematics education in schools should merge with education in computer science / programming. Of course, this means a deep structural reform of school mathematics education. We are not talking about adapting the 20th century mathematics to the 21st century—as it outlined in [10, 19], we mean the 21st century mathematics education for the 21st century mathematics. To the best of our knowledge, this paper is perhaps the first known attempt to start a proper feasibility study for this reform. The scope of the paper does not allow us to touch the delicate socio-political (and financial) sides of the reform, we are looking only at general curricular and didactic aspects and possible directions of the reform. In particular, we indicate approaches to development of a Domain Specifiic Language (DSL) as a basis for all programming aspects of a new course.

1. Executive Summary

This text discusses some aspects of curriculum for a proposed replacement of the traditional school algebra course by a new course which fully integrates algebra with informatics / programming. Approaching this issue from the basic principles of project management, we focus on two points from which development of any project starts:

(1) What are our aims? What do we wish to achieve?
(2) What is our starting position?

In the context of curriculum development, this becomes

Question 1: What are learning outcomes of the new course? What school graduates should be able to do on completion of the course?

Question 2: How the new course will relate to the primary school arithmetic?

In case of projects directed at work with children for considerable periods of time (in the case of algebra curriculum, this is perhaps 7 or even more years), point (2) and Question 2 attain larger importance than, say, in most industrial projects: the child has to be in the focus of the project, and the child grows, and all the proposed activities have to grow with him.

Date: November 22, 2022.

Both points need to be clearly described and illustrated by sample problems of the kind Learner is supposed to be able to solve (on their own, without guidance from teachers) at the start and at the end of their study. Only then we can start writing a curriculum, working in the direction from the answer to Question 1 to answer to Question 2, at every step including only the material which is definitely needed for further progression to a higher level.

1.1. **Regarding Question 1, we formulate the following recommendations.**

1.1.1. The most important issue of the 21st Century is the relationship between people and computers.

Therefore we expect that on completing the course, Learner should

- have understanding of opportunities and dangers coming from saturation of every aspect of the economy, politics, warfare, media, entertainment, everyday life, by computers and computer systems which make decision on behalf of people;
- in navigating this dangerous world, understand what questions could and should be asked—and to whom;
- if necessary, be prepared to learn, and use, more technical and professional aspects of computer science and computer programming.

This could be compared with mathematical skills expected from secondary school graduates in Europe in the first half of the 20th century: they had to master enough of algebra and trigonometry for their subsequent training, if needed, as artillery officers or air pilots—after all, it was the era of mass conscription armies.

1.1.2. There is a simple and challenging criterion of Learner's the mastery of the required skills.

- From a relatively early stage of the course, Learner's answer to an arithmetic or algebraic problem (including mathematical problems described as 'real life' problems) should be an executable computer code developed, almost in its entirety, by Learner—without use of standard packages such as MATHEMATICA—which
  - solves *all* problems of the same type;
  - helps to check, analyse, and generalise the solution.

  Problems which require a *proof* of some statement are naturally excluded from this requirement—but computer experiments should be welcome in construction of counterexamples and in formulation of conjectures.

1.1.3. The requirements for a computer language to be used in the course are quite demanding.

- If more advanced versions of it are used at later stages of learning, it needs to be backward compatible at all stages of school algebra—to allow a systematic revisiting, reusing, and rethinking of earlier learned material, problems solved, and codes written.
- But its small "starting" fragment should provide a simple, safe, and exiting playground for children at the first steps of learning algebra.
- The language should include sufficiently rich elements usually found in high level languages and provide efficient preparation for learning professional industry standard languages.

- All numbers have to be exclusively symbolic, including fractions and surds; if the result in some intermediate calculation is $\frac{1-\sqrt{3}}{2}$, it stays that way and is used in subsequent calculation. Of course, they have to include constants e and π. No rounding, no floating point decimals.[1]

This will require development of a bespoke *Domain Specific Language*[2] (DSL) for use in the course. Most likely, existing general purpose languages are not suitable for this role.

This texts represents first early steps in development of specifications for DSL.

1.1.4. The content of the algebra course should change and reflect the demands of computer programming, and include, for example, Boolean algebra, elements of number theory, modular arithmetic, and finite fields. Shaping the algebraic curriculum should go in step with development of the DSL.

1.1.5. Other parts of mathematics, for example, geometry, mechanics, and statistics need a separate discussion and are not touched here. Also, interactions with physics deserve most serious attention.

1.2. **On Question 2, our principal suggestions are the following:**

- Introduction of "computational" thinking in algebra should be prepared by development of algorithmic thinking in arithmetic.
- "Questions method" for word problems in arithmetic is suggested as an efficient tool for mastering an elementary level of algorithmic thinking—and this has done in arithmetic before moving to algebra.
- Use of a simple computer language (which still needs to be developed) is essential from the first day of learning algebra, in particular, for formulation of algorithms developed by the arithmetic 'questions method' in a compact form suitable for conversion into computer code.
- Typed variables and type inference are essential for a computer language that will be used in the course, and their introduction needs to be prepared by mastering 'named numbers' in arithmetic. It needs to be clarified that

  named numbers are ***not*** *numbers replaced by names and symbols,*

  they are numbers of units of various kind: 10 apples are not the same as 10 people and not the same as 10 kilometers, and you do *not* add 10 apples and 10 people, see Figure 5.
- However, there is a conceptual gap between
  - formulation of an algorithm and its implementation in a code, and
  - representation of the answer as a closed algebraic formula.

  Therefore Learner will still need mastering algebraic manipulations with formulae in algebra—which is conceptually parallel to use of evaluation by call by name in computer programming—as opposed to evaluation by call by value used in a direct implementation of an a arithmetic algorithm.

[1] And the answer to the question "Find the reminder upon division of polynomial $x^{2023}+1$ upon x^2-4" should be $2^{2022}\cdot x+1$, without any attempt to print 2^{2022} in decimals.

[2] From Wikipedia: "A Domain-Specific Language (DSL) is a computer language specialized to a particular application domain. This is in contrast to a general-purpose language (GPL), which is broadly applicable across domains." Famous examples include html and LaTeX. Some books: [3, 18, 24]. We are using Wikipidea as a primary reference because neither us, nor our expected readers are computer scientists or computer programmers.

- A lot of attention needs to be given to *arithmetic of named numbers*. In particular, Learner should learn that they can freely introduce in their solutions intermediate parameters, and their values, if the names of these parameters (correspondingly, types of variables) are alien, appear neither in the data (conditions) of the problem, no in the supposed answer. These additional parameters will disappear in the course of solution.
- This approach inevitably requires Learner reflect on his/her work, analyse it, apply, in effect, some meta-thinking (that is, thinking about thinking)—and can be achieved only if Learner is psychologically comfortable and feels being in full control of the problem, its solution, and the computer.

> Achieving all that is a challenging task. We would not advise to undertake it lightly.

1.3. **Warning about the scale and cost of the project.**

The first author has experience of working on large and complex research projects, see for example [1], a book of 550+ pages that he co-authored and which contained a proof of a single theorem—the proof was in making for about 15 years and used crucially important contributions from an informal team of 10 people. Closer to the theme 'algebra + computer science for school' is his work on computational symbolic logic, where his joint paper with Şükrü Yalçınkaya [11] was developed, step by step, over at least 10 years, and progress critically depended on systematic computer experiments.

Basing his assessment on this experience, he is quite confident that development of a software system supporting the proposed course and *used on the scale of a nation* is realistic, but could easily cost up to a few million dollars, demand work of a large interdisciplinary team of experts, and, with necessary test runs in schools, could take up to ten years. This estimate does not include the cost of re-education of the whole army of teachers, and time required for that.

We urge the reader to remember the words of H. L. Mencken:

> *For every complex problem there is an answer that is clear, simple, and wrong.*

We warn:

> Beware of snake oil merchants peddling cheap and easy recipes for revitalising school mathematics by means of 'digital technologies'!

We hope that our work could be used as an antidote against their promises.

***

The rest of the paper contains a more detailed discussion of Question 2:

How the proposed course will relate to the primary school arithmetic?

2. But what is arithmetic?

School arithmetic contains two major parts.

- Formal written methods for arithmetic operations on decimals and fractions [15], that is, manual computation based on certain recursive algorithms.
- Solving word problems.

As explained in [6], "word problems" of arithmetic involve identification of mathematical structures and relations of the real world and mapping them onto better formalised structures and relations of arithmetic, or, in Igor Arnold's formulation from 1946, when the words 'structure' and 'relation' were not yet *en vogue* [26],

> *... teaching arithmetic involves, as a key component, the development of an ability to negotiate situations whose concrete natures represent very different relations between magnitudes and quantities.*

Even more important is Igor Arnold's characterisation of arithmetic:

> *The difference between the "arithmetic" approach to solving problems and the algebraic one is, primarily the need to make a concrete and sensible interpretation of all the values which are used and/or which appear at any stage of the discourse.*

Therefore the phase transition from algebra to arithmetic is an important turning point in learning mathematics.

3. The bridge from arithmetic to algebra

This is our principal thesis:

> The merger of school mathematics with informatics should start simultaneously with the phase transition of mathematical learning from arithmetic to algebra, and should be rooted in those aspects of arithmetic which actually belong to computer science but are not usually recognised as such.

For example, arithmetic already contains

Abstraction: The concept of *number* is already a huge abstractio.

Algorithms: First of all, long division, etc. are algorithms. But they are given to children as something God-sent, just as rules to follow. However, what will be shown here, the 'questions method' of yesteryear arithmetic (now almost universally forgotten (one of its rare clear exposition is [28]) allows children to develop their own algorithms which solve many types of arithmetic problems.

Recursion: The notorious long division is a recursive algorithm, as well as long multiplication, as well as addition and subtraction of decimals. Euclid's algorithm for finding the greatest common divisor of two integers is also recursive.

Typed variables: 'Named numbers' of arithmetic perfectly map to typed variables of computer programming.

Reification: This is a high level concept from computer science, but its toy version appears in arithmetic as introduction of intermediate parameters ('helpful numbers' is a better term for kids' use), see Section 5.5. Or as

'fantasy units of measurement' – see [12, Sections 3.1 and 3.3] and [9, Solutions and Notes to Problems 89 and 90] (in the latter they are called 'hidden parameters') for discussion and historic examples.

In computer science, *reification* is a widely used concept. This is how WIKIPEDIA defines reification:

> *Reification* is the process by which an abstract idea about a computer program is turned into an explicit data model or other object created in a programming language. A computable/addressable object – a resource – is created in a system as a proxy for a non computable/addressable object.[3]

In school algebra, the notorious x as a notation for the unknown is a typical example of reification: it is a proxy for a number not given to us, but a proxy nevertheless allowing algebraic manipulations with it.

The rest of this text concentrates on this issue: the "questions method" and algorithms in transition from arithmetic to algebra. Its principal point is

> Fusion of algebra and computer science should be prepared by development of algorithmic thinking in arithmetic.

We shall explain, that, in school arithmetic of yesteryear, there was an approach to an efficient development of algorithmic thinking: the so-called "questions method" for solving word problems[4]. We further claim that it could serve as a bridge between arithmetic and algebra, especially if algebra is merged with computer science / coding.

We shall also try to demonstrate that the questions method allows us to reach a number of didactic targets:

- Emphasis on detection and analysis of structures and relations of the real world and their representation in mathematical terms.
- Systematic analysis of data given in a problem.
- Correspondence between named numbers in algebra and typed variables in code writing.
- An answer to an arithmetic or algebraic problem being an executable computer code which
  - solves *all* problems of the same type;
  - helps to check and analyse the solution;
  - uses evaluation by call by name and by call by value.

If these targets are achieved, we will be able to claim that

> The child is in control of the problem, of its solution, and of the computer [5].

And we have to achieve that—otherwise, no further progress is possible.

We do not discuss here why a fusion of algebra with computer science could be desirable in school education – this was discussed in [10] and [19].

[3] `https://en.wikipedia.org/wiki/Reification_(computer_science)`.

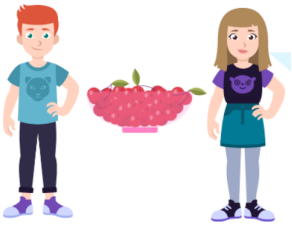
[4] The questions method is discussed in some details in [6, Sections 3 and 4].

4. A sample problem from arithmetic: Bowl of cherries

Let us consider a simple problem.[5] Of course, at school children should start with much simpler, 1 or 2 step problems; problems that the one below should come at a later stage. We use this example because it provides a wider overview of the arithmetic / algebra boundary.

Bowl of cherries
Alice and Bob pick cherries.

Bob: I can fill this bowl in 8 minutes.



Alice: I can fill this bowl in 24 minutes.

Bob Alice

Working together, in what time they will fill the bowl?

Let us approach this problem step by step in compliance with the methodology that we have just formulated.

5. Analysing data

5.1. **Parsing of the problem.** Let us now to ask the all-important question:

What is the data and what is the question?

Indeed, this question is well justified because the correct parsing of the problem is the following:

> **Data:** Bob fills the bowl in 8 `min`, Alice in 24 `min`.
> Now they work together.
> **Question:** In what time they will fill the bowl?

We shall call the question in the problem that we separated from the data the *target question*.

[5]The problem, and the picture illustrating it, was picked from an online talk by Olga Moskalenko «Проблема систематических ошибок в решении текстовых задач по результатам мониторинга на платформе «Учи.ру»» on Научный семинар по методике преподавания математики, Moscow State University, 22 September 2022. Interestingly, in a test only 39% of Year 6 students gave the correct answer, 6 minutes.

5.2. **Questions method and step questions.** And now we shall give an example of a questions method. In the 1960s in Russia, students were supposed to master it by the age of 11 or 12 and be able to produce, for a word problem, a short sequence of simpler *step questions*, which resulted in answering the target question. This could also be described as a formulation of a multi-step solution to a problem as a composition of solutions to much simpler one step problems.

One of many possible step question sequences for the BOWL OF CHERRIES problem is shown in Section 6.1. We shall try to explain now how it could be obtained.

5.3. **Generic guiding questions.** Teaching children to formulate step questions in a questions method focuses on development of each child's ability to start his/her "questions" attempt at a solution by asking himself or herself appropriate self-directed *guiding* questions (they were called *auxiliary questions* in the Russian pedagogical literature, but in England, the words "*an auxiliary question*" are loaded with expectation that the question is asked by a teacher to help a struggling pupil).

Generic questions (or meta-questions) recommended by the questions method could be data-centered [25, 27, 28, 29], for example,

- What questions can be asked about these data?[6]
- What can be learned from these data?
- And a more specific question: how to express mathematically "working together"?

Beloshistaya [28, p. 303] calls this approach *synthetic*. It is interesting to compare Beloshistaya's analysis with Gerofsky [17].

There are also more advanced guiding questions. Those found in the modern literature appear to come directly from a project management manual [29].

> *Analyse the task working backwards from the target back to the current situation.*

Indeed look for yourself at questions of the synthetic approach:

- What additional quantities it would be useful to know for finding the answer? This question could lead to introduction of a 'helpful number', see Section 5.5.
- What do you have to do for finding these additional quantities?
- How can they be deduced from the data given?

5.4. **The crucial point: To whom these questions were addressed?** In the old tradition of the "questions method",

- At the first stages of learning the questions method, guiding questions were addressed by a teacher or a textbook to **students**, aged 11–12.
- At later stages, **students** were expected to formulate there guiding questions **themselves**, thus developing **their own step questions** directed at solving a specific problem.

[6] In the 1960s in Russia these questions could be asked even in a harsher form: a teacher wrote the question on the blackboard, gave a minute to students to think, then wiped out the target question and asked: what questions could be asked about these data? What could be learnt from these data?

- Put together, **student's** step questions formed in fact an **algorithm** (although this word was not mentioned): any problem of the same type, but with different data, could be quickly answered by following the same steps.
- This algorithm had to be developed by the student.

5.5. **Introduction of an intermediate parameter (or a "helpful number").** Answering a guiding question

> How to express mathematically "working together"?

after some thought or discussion it could be discovered that "to work together" means, in mathematical terms, that

> productivities, or speeds of picking cherries, *add up*.

This should immediately trigger further questions:

- How to measure the speed of picking?
  Of course, in *cherries per minute*!
- How many cherries we have in the problem?

And the last question leads to a *critically important step*:

A helpful number: Let us assume that the bowl contains 72 cherries.

This choice, 72, appears to be arbitrary and made just for the convenience of division by 8 and by 24, which allows to avoid fractions in further calculation. We shall soon see that this is well justified. Also, as it has already been mentioned in Section 3, it is an example of *reification*, one of concepts of computer science already present, in disguised or hidden form, in arithmetic. Later, in Section 6.2, it will be explained why this step is safe.

6. Solution

6.1. **The sequence of step questions.** The following solution is a sequence of questions and answers formulated on the basis of analysis of the data and in response to generic guiding questions described in Section 5.

Question 1. What is Alice's picking speeds?

$$3\frac{\texttt{cherries}}{\texttt{min}} = \frac{72\ \texttt{cherries}}{24\ \texttt{min}}$$

Question 2. What are is Bob's picking speed?

$$9\frac{\texttt{cherries}}{\texttt{min}} = \frac{72\ \texttt{cherries}}{8\ \texttt{min}}$$

Question 3. What is their picking speed working together?

$$12\frac{\texttt{cherries}}{\texttt{min}} = 3\frac{\texttt{cherries}}{\texttt{min}} + 9\frac{\texttt{cherries}}{\texttt{min}}$$

Question 4. In what time they will fill the bowl?

$$6\ \texttt{min} = 72\ \texttt{cherries} \div 12\frac{\texttt{cherries}}{\texttt{min}}$$

6.2. **Elimination of intermediate parameters of independent types.** But what will happen with the answer to the problem if we change the number of cherries—it was not given to us, it was selected by us just for convenience?

Nothing will happen.

The data given is of type `min`, the answer is also of type `min`. The extra parameter is of completely independent type `cherry`. The answer does not depend on the numerical value of the extra parameter, because its change can be treated simply as the change of *unit of measurement*: instead of 72 cherries we can talk about 32 `pairs`, or 24 `spoons`, or 12 `cups` of cherries, etc.—all these different numbers describe the same physical quantity.

This simple observation gives us

> **Freedom of choice of intermediate parameters** — we need only to make sure that their types are independent from types of data.

6.3. **The resulting (pseudo)code.** This is the setup of the problem which uses letters instead of numbers:

> Bob fills the bowl in `B min`, Alice in `A min`
> Working together, they fill the bowl in `T min`.

A "questions method" solution can be easily rewritten as a pseudocode:

```
input A, B                                  % Alice's and Bob's times
input C                                         % number of cherries
U := C:A;   V := C:B;                      % Alice's and Bob's speeds
W := U + V;                 % Their picking speed working together
T := C:W                    % The time in which they fill the bowl
return T;
```

But how to simplify it to a one line pseudocode:

```
input A, B;   T:= (A*B):(A+B);   return T
```

that is,

$$T := \frac{AB}{A+B}?$$

This is a serious issue.[7] In the early algebra, writing code is **much** easier than developing an algorithm. Indeed the answer

$$T := \frac{AB}{A+B}$$

—the shortest expression of the algorithm —is non-trivial.

[7] In the discussion of this problem at the Association of Teachers of Mathematics (UK) meeting in October 2022, one of my colleagues quite rightly asked "I am wondering how a student might become aware that $(AB)/(A+B)$ might be a solution to the problem?" And someone else suggested "I think of it as $1/(1/A+1/B)$ rather than it's simplified form".

6.4. **Solution in symbolic variables.** Compression of the code obtained by the questions method into a compact formula could be done either manually, or maybe even automatically; for the latter case, the code has to be embedded into a symbolic algebra system which automatically simplifies algebraic expressions. We think it would be desirable to achieve a didactically efficient balance of the two approaches.

Setup: A: Alice's speed, B: Bob's Speed, C: number of cherries.

Question 1. What are Alice's and Bob's picking speeds?

$$\frac{C}{A} \quad \text{and} \quad \frac{C}{B}$$

Question 2. What is their picking speed working together?

$$V := \frac{C}{A} + \frac{C}{B}$$

Question 3. In what time they will fill the bowl?

$$T := \frac{C}{V} == \frac{C}{\frac{C}{A} + \frac{C}{B}} == \frac{AB}{A+B}$$

Notice that this is *"call by name"* evaluation.

Also notice that variable C disappeared from the answer—as it was inspected, because it was of type independent from the type of data.[8]

6.5. **A completely different solution.** Of course, the questions method can produce other solutions as well. Here is one of them.

Question 1. In how many times Bob is faster than Alice?

$$\frac{24\,\mathtt{min}}{8\,\mathtt{min}} = 3 \quad \text{or} \quad \frac{A}{B}.$$

Question 2. In how many times the two of them working together, are faster than Alice working alone?

$$3 + 1 = 4 \quad \text{or} \quad \frac{A}{B} + 1.$$

Question 3. In what time they will fill the bowl?

$$24\,\mathtt{min} : 4 = 6\,\mathtt{min} \quad \text{or} \quad \frac{A}{\frac{A}{B}+1} = \frac{AB}{A+B}.$$

[8]In an earlier discussion of this text, there was an interesting question about this problem: "Who ate all the cherries?". Well, we can add to the setup of the problem someone called Kevin, who eats cherries at the rate of 1 bowl per K minutes. The code could be easily adjusted producing the answer that the bowl will be filled in time

$$\frac{ABK}{AK + BK - AB},$$

and also providing a warning that if

$$K \leqslant \frac{AB}{A+B},$$

the bowl will never be filled. However, some colleagues suggested, quite rightly, that the equivalent inequality which characterises the failure to fill the bowl

$$\frac{1}{K} \geqslant \frac{1}{A} + \frac{1}{B}$$

was easier to interpret in real life terms.

7. User's Story

We hope that this "questions method" solution can be naturally converted into a computer code with the help of a child friendly GUI interface. What follows is a brief discussion of a desirable functionality of this interface, a "User's Story", in terminology of software development, written in accordance with recommendations of *User experience design*[9].

Assuming that a student (we call him/her Learner) types everything in some computing device (smartphone, tablet, laptop) with the course software installed, as soon a number is entered, the GUI starts a dialogue with Learner asking whether this number is a datum, or a helpful number, or the answer to the problem and what is its type. Also, in this dialogue a number of various ambiguities are resolved—we omit these details here. In simple problems, this dialogue is likely to be very short—we shall explain in the next our paper.

The GUI assigns internal variables to all numbers on the screen, and converts numbers into cells whose values could be changed by the code or edited by Learner The values of data and helpful numbers remain editable by Learner see Figure 1. It is an executable code represented in a GUI. The 'Hamburger' icon ☰ is a pop-up menu with many additional functions.

If Learner chooses button Run in the menu and presses it, the cells become alive and the GUI looks as in Figure 2.

The Learner is invited to experiment with changing data and auxiliary parameters; it the helpful number is changed from 72 to 48, and the button Run is pressed, the Learner gets another screen, Figure 3—with exactly the same answer!

Even more important: it should be possible to substitute letters in the cells (or in just one cell) for data and helpful numbers, thus automatically getting a symbolic solution of the kind offered in Section 6.4. See Figure 4 for what happens if the time for Alice to fill the bowl is denoted by letter A.

The course software should help Learner to move from arithmetic to algebra: a solution of an arithmetic problem can be converted into a code which solves the same problem, but with symbolic inputs.

8. Named numbers

8.1. **Types of named numbers.** Merging Algebra with Computer Science requires development of an appropriated Domain Specific Language which should support calculations with *named numbers*, of the kind as in the situation when we *share* 10 apples among 5 people:

$$10\,\texttt{apples} \div 5\,\texttt{people} = 2\frac{\texttt{apples}}{\texttt{people}}.$$

Obviously, we need a typed language where variables could be of arbitrary type assigned to them, say `apple`, or `people`, where only variables of commensurable (or comparable) types could be added, say $1\texttt{m} + 1\ \texttt{cm} = 101\ \texttt{cm}$, but not apples and people (see Figure 5); this restriction is possible to achieve[10]; however, division of

[9] User experience design, `https://en.m.wikipedia.org/wiki/User_experience_design`.

[10] See Sirotin [22], a library of functions and objects of programming language Kotlin (`https://kotlinlang.org/`) which allows working with variables whose values are expressed in the International System of Units SI like meter, second etc. as well as some other common units like currencies, percentages etc..

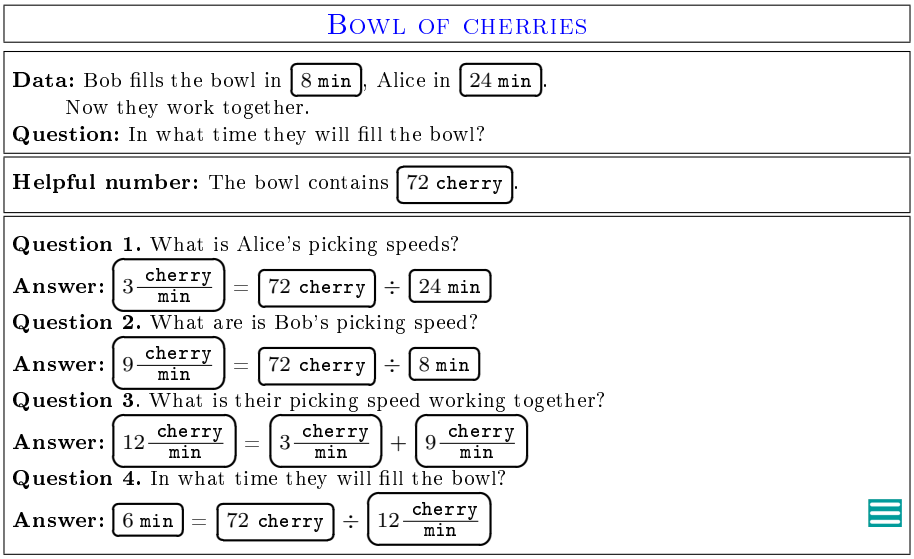


Рис. 1. GUI after entering the solution. The 'Hamburger' icon is a pop-up menu with very additional functions.

`apples` by `people` automatically produces a variable of a different type, $\frac{\texttt{apples}}{\texttt{people}}$. Some types, such as SI units or currencies, need to be standard, built in the DSL—but Learner has to be able to create any types of fancy, say `ducat` or `piastre` and declare exchange rate $1\,\texttt{ducat} == 5\,\texttt{piastre}$, so the types `ducat` and `piastre` become commensurable.

A more sophisticated manipulation with types is needed when we *dispense* 10 apples, 2 apples a person, and wish to know how many people will get their apples:

$$10\,\texttt{apples} \div 2\,\frac{\texttt{apples}}{\texttt{people}} = 5\,\texttt{people}.$$

8.2. **A bit of history.** Actually, it was clearly understood by the father of modern (symbolic) algebra, François Viéte who in 1591 wrote in his *Introduction to the Analytic Art* [23, p. 16] that

> If one magnitude is divided by another, [the quotient] is heterogeneous to the former ... Much of the fogginess and obscurity of the old analysts is due to their not paying attention to these [rules].

Alas, these words remain true in the XXI Century.

This is the dirty secret of school arithmetic: it lives in an intimate relationship with the arithmetic of *types* which is carefully hidden from children (and from many teachers). However, this is well known in physics where types are called *dimensions*, and where *dimensional analysis* is a simple, but powerful method of understanding of relations between quantities and magnitudes of different nature, which, in particular, gives a way to produce quick, frequently even quick *mental* estimates of magnitudes. For example, [4, Section 8.4] contains a one page deduction of the legendary Kolmogorov's "5/3" law for the energy spectrum of turbulent movement of gas or liquid.

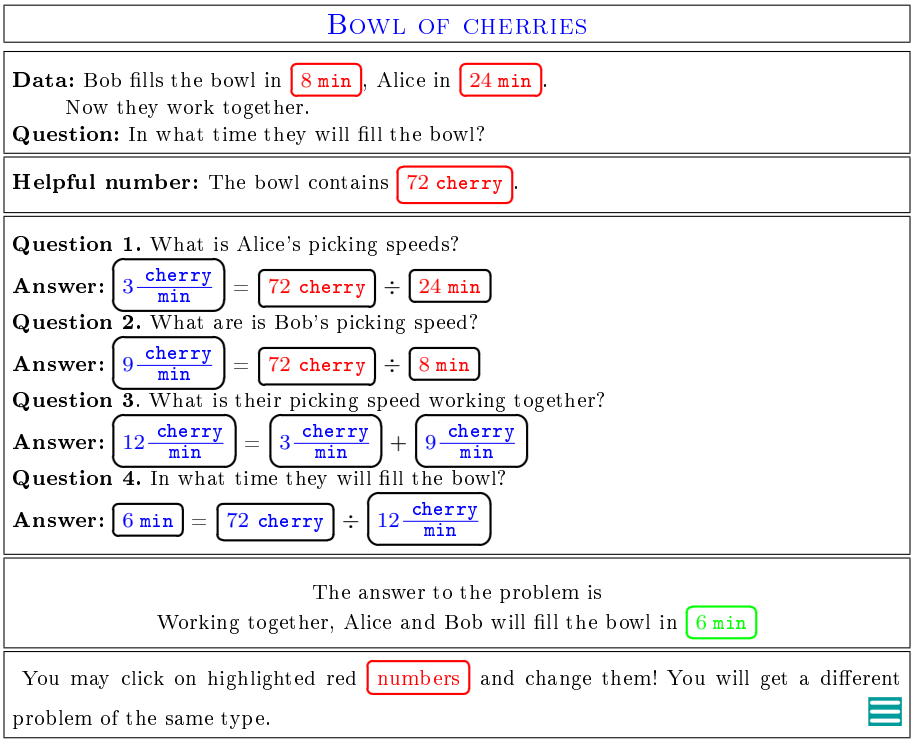


Рис. 2. The code goes alive. This is something that could be proudly shown to a parent.

8.3. **An elementary example from physics.** Let us look at one of simpler applications of dimension (type) analysis in physics.

Galileo Galilei observed that the period of pendulum does not depend on the amplitude (span) of its movements.

So, the period T (of type `second`) of pendulum depends on its length L (of type `meter`) and acceleration of gravity g (of type $\frac{\texttt{meter}}{\texttt{second}^2}$).

The only type-consistent formula which can be made out of that is

$$T = C\sqrt{\frac{L}{g}},$$

where C is dimensionless constant. Even if we do not know the value of C (this requires more subtle arguments), we may draw very interesting consequences.

For example, bipodal walking can be modeled as a sequence of falls in which a leg (of length L) which moves forward behaves as a pendulum of period T. Hence the speed V is proportional to $\frac{L}{T}$ and

$$V \sim \frac{L}{T} \sim \frac{L}{\sqrt{\frac{L}{g}}} \sim \sqrt{gL}.$$

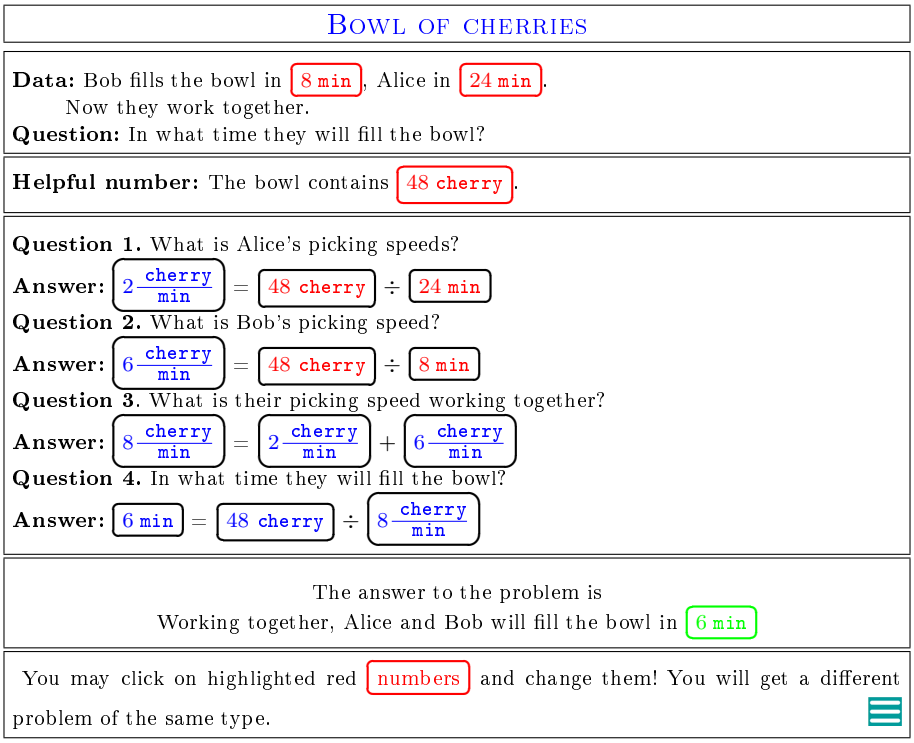


РИС. 3. Manipulation with 'helpful number': 72 `cherry` replaced by 48 `cherry`—the answer did not change.

Hence walking on stilts increases speed (anyone who tried to walk on tilts knows that), Figure 6, while walking on the Moon is $\sqrt{6}$ slower than on Earth (since the acceleration of gravity on the Moon is about $\frac{1}{6}g$).

Dimension analysis should be part of the school course of Physics. It is simple, beautiful, and has a potential to produce revelations.

8.4. **Type analysis in arithmetic.** Let us apply the type analysis to a classical old problem, a part of mathematical folklore:

> RABBITS AND CHICKEN. Mary has pets, some rabbits and some chicken. Together her pets have 12 heads and 32 legs.
>
> How many rabbits and how many chickens does Mary have?

It is more open to type analysis because it obviously involves more than one type.

First of all, we have to carefully assign types to every piece (datum) of quantitative data that we are given, or know from out lived experience.

Of course, we are given data of types `leg` and `head`. So, we have to introduce variables LEGS and HEADS of these types, respectively, and we are given their values.

Since every rabbit and every chicken has exactly one head, we have variables RABBITS and CHICKEN which could be safely thought of being of type `head`.

Bowl of cherries

Data: Bob fills the bowl in $8\ \mathtt{min}$, Alice in $A\ \mathtt{min}$.
Now they work together.
Question: In what time they will fill the bowl?

Helpful number: The bowl contains $1\ \mathtt{cherry}$.

Question 1. What is Alice's picking speeds?
Answer: $\frac{1}{A}\ \frac{\mathtt{cherry}}{\mathtt{min}} = 1\ \mathtt{cherry} \div A\ \mathtt{min}$
Question 2. What is Bob's picking speed?
Answer: $\frac{1}{8}\ \frac{\mathtt{cherry}}{\mathtt{min}} = 1\ \mathtt{cherry} \div 8\ \mathtt{min}$
Question 3. What is their picking speed working together?
Answer: $\frac{A+8}{8A}\ \frac{\mathtt{cherry}}{\mathtt{min}} = \frac{1}{A}\ \frac{\mathtt{cherry}}{\mathtt{min}} + \frac{1}{8}\ \frac{\mathtt{cherry}}{\mathtt{min}}$
Question 4. In what time they will fill the bowl?
Answer: $\frac{8A}{A+8}\ \mathtt{min} = 1\ \mathtt{cherry} \div \frac{A+8}{8A}\ \frac{\mathtt{cherry}}{\mathtt{min}}$

The answer to the problem is
Working together, Alice and Bob will fill the bowl in $\frac{8A}{A+8}\ \mathtt{min}$

You may click on highlighted red numbers and change them! You will get a different problem of the same type.

Рис. 4. Substitution of a letter for a datum.

Not much so far. Perhaps we have to turn to our lived experience: comparative anatomy of rabbits and chicken. Perhaps it means

$$\textsc{RabbitAnatomy} = 4\frac{\mathtt{leg}}{\mathtt{head}}$$

and

$$\textsc{ChickenAnatomy} = 2\frac{\mathtt{leg}}{\mathtt{head}}$$

So all data that we have are

$$\begin{aligned} \textsc{Heads} &= 12\,\mathtt{head} \\ \textsc{Legs} &= 32\,\mathtt{leg} \\ \textsc{RabbitAnatomy} &= 4\frac{\mathtt{leg}}{\mathtt{head}} \\ \textsc{ChickenAnatomy} &= 2\frac{\mathtt{leg}}{\mathtt{head}} \end{aligned}$$

We have to understand, that for a child, this analysis of data is challenging – but exactly this skill,

ability to see mathematical structures and relations in the real world

– is missing in the mainstream mathematics education. But it has to be taught, and systematically.

PHC. 5. The First Law of Arithmetic: you do not add fruit and people. Giuseppe Arcimboldo, *Autumn*. Wikipedia Commons. Public domain.

Taking into account François Viéte's advice, and the natural limitations of type arithmetic, we have *very little choice* of sensible arithmetic operations between these values, we emphasize, *we have very little choice*. One of the *very few* actions that we may try is to compute

$$\textsc{Heads} \times \textsc{RabbitAnatomy} = 12\,\mathtt{head} \times 4\frac{\mathtt{leg}}{\mathtt{head}} = 48\ \mathtt{leg}.$$

What we see here is *type inference*, in terminology of computer science. But does it have a real life meaning? Yes, it does. This is the number of legs Mary's pets would have it all of them were rabbits. This is more than the given number of legs. By how much more?

$$\textsc{ExcessiveLegs} = \textsc{Heads} \times \textsc{RabbitAnatomy} - \textsc{Legs} = 16\,\mathtt{leg}.$$

Where do these excessive legs come from? From chicken, each getting extra

$$\textsc{RabbitAnatomy} - \textsc{ChickenAnatomy} = 4\frac{\mathtt{leg}}{\mathtt{head}} - 2\frac{\mathtt{leg}}{\mathtt{head}} = 2\frac{\mathtt{leg}}{\mathtt{head}}$$

Рис. 6. Walking on stilts increases speed. *Habitants des Landes.* Jean Louis Gintrac (1808–1886). Source: Wikipedia. Public domain.

legs. So, what is the number of chicken?

$$\begin{aligned}\textsc{Chicken} &= \frac{\textsc{ExcessiveLegs}}{\textsc{RabbitAnatomy} - \textsc{ChickenAnatomy}}\\ &= 16\,\mathtt{leg} \div 2\frac{\mathtt{leg}}{\mathtt{head}}\\ &= 8\,\mathtt{head}\end{aligned}$$

Of course, the number of rabbits after that is obvious:

$$\textsc{Rabbits} = \textsc{Heads} - \textsc{Chicken} = 12\,\mathtt{head} - 8\,\mathtt{head} = 4\,\mathtt{head}.$$

One of the *very few* other available options to start our solution could be

$$\textsc{Heads} \times \textsc{ChickenAnatomy} = 12\,\mathtt{head} \times 2\frac{\mathtt{leg}}{\mathtt{head}} = 24\ \mathtt{leg},$$

but it leads to essentially the same solution.

This is not the shortest way to solve this problem but it has an advantage of being very formal, with every step forced on us, and leading to a computer code manipulating with named numbers – this solution is already almost a code.

Alternative approaches are possible. Recently, the first author had one of his regular chats on Google Meet with a small group of Ukrainian refugee children (of ages about 8 to 11), and discussed with them this problem, and one of the kids suggested to look at `extremity`, or a limb, that is, `leg` or `wing`, which leads to a more efficient solution – but only because a deeper insight in the real world is applied, and more types are used. Indeed, just three straightforward questions:

(1) What is the number of extremities?
(2) What is the number of wings?
(3) What is the number of chicken?

lead to a solution.

9. We need fusion of algebra and computing, not concatenation of the two

A paper by Ian Benson and Jim Thorpe [2] advertises the old paper by Trevor Fletcher [14] *Thinking with Arrows* as 'conceptual mathematics', and a way to 'represent and reason' about mathematical patterns.

Perhaps driven by the law of nominative determinism, Fletcher was enthusiastic about use of arrows in school mathematics. This is a fragment from his paper given by Benson and Thorpe as an example of 'thinking with arrows'.

> If you have a tap that fills a bath in 3 minutes and a tap that fills the same bath in 6 minutes, you have to know how to decide that together they would take 2 minutes. Likewise if you are told 10 minute and 10 minutes you have to be able to reply 5 minutes. How is the calculation done? This time the operation of getting 2 from 3 and 6 may be called *tap*, and the associated diagram is

(1)

$$\begin{array}{ccc} (3,6) & \xrightarrow{\;tap\;} & 2 \\ {\scriptstyle recip}\,\vdots & & \vdots\,{\scriptstyle recip} \\ \left(\frac{1}{3},\frac{1}{6}\right) & \xrightarrow{\;+\;} & \frac{1}{2} \end{array}$$

Here, we do not see how the answer to the problem was **found**; we see only that the **answer** to the problem is **represented** by a commutative diagram which reduces the answer (treated as a binary operation) to the binary operation of addition of positive real numbers; please notice that the word 'operation' is used by Fletcher.

We have objections to this approach.

- We think the aim of mathematics education should be teaching the child the art of **solving** problems and **getting answers**.
- **Interpretation** of answers should play important role; however, insistence on a particular form of **representation** of the answer is undesirable; after all
- There should be a smooth transition between the arithmetic / algebraic thinking in solving a problem and computer science / computer programming thinking in representing the solution as an algorithm.
- We need fusion of algebra and computing, not concatenation—but Diagram 1 is a concatenation of two disjoint conceptual areas.

We wish to demonstrate, that, in the case of a class of problems on the arithmetic / algebra boundary where the tap problem naturally belongs, the commutative diagram representation of a solution is misleading and counterproductive.

Indeed, consider the 'tap problem' together with three other problems from the same stable: proportionality dependencies between time, speed, and distance (where 'distance' could perhaps be 'level of water in the bath').

> **Problem 1:** If you have a tap that fils a bath in a minutes and a tap that fills the same bath in b minutes, in how many minutes they would fill the bath

together? This is, of course, Fletcher's problems with answer represented in Diagram 1.

Problem 2: If a car travels from A to B at speed of a miles per hour, and then returns from B to A at speed of b miles per hour, what is the average speed of the car on the whole journey?

Problem 3: Two cars started their journeys at sunrise, one from A to B, and another one from B to A. They met at noon and completed their journeys at a hours p.m. and b hours p.m., correspondingly. By h ow much was sunrise earlier than the noon on that day?

Problem 4: There are two cities on a river, one is upstream of another. It takes for a steamboat a days to get from one city to another, and b days to return back. How many days it will take a raft to drift from the city which is upstream to the downstream city?

We give here answers to these problems in random order:

$$\text{(a) } \sqrt{ab};\ \text{(b) } \frac{ab}{a+b};\ \text{(c) } \frac{2ab}{|a-b|};\ \text{(d) } \frac{2ab}{a+b};$$

How would the child know which answer is for which problem? Being told by the teacher? Where do these answers come form? Fletcher's commutative diagram is indeed commutative, but, we are afraid, this is a bad example of theoretical-methodology-driven approach to teaching, not a child-focused approach to child's learning. On the contrary, we suggest, for the critical stage of moving from arithmetic to algebra, a smooth way from an algorithmic solution of an arithmetic problem to its algebraic expression and coding. Commutative diagrams could come into the play a few years later, not now.

If we go into the underlying mathematics, we quickly discover that only one of these answers (a)-(d) can be represented by a commutative diagram similar to Diagram 1. Indeed, if $\circ$ and $\diamond$ are two binary operations on the set of positive real numbers $\mathbb{R}^{>0}$ satisfying the commutative diagram

(2)
$$\begin{array}{ccc} (a,b) & \xrightarrow{\ \circ\ } & a\circ b \\ {\scriptstyle\phi}\,\vdots & & \vdots\,{\scriptstyle\phi} \\ (\phi(a),\phi(b)) & \xrightarrow{\ \diamond\ } & \phi(a)\diamond\phi(b) \end{array}$$

for some map $\phi : \mathbb{R}^{>0} \longrightarrow \mathbb{R}^{>0}$, then

$$\phi(a\circ b) = \phi(a)\diamond\phi(b), \tag{3}$$

that is, ϕ is a homomorphism from monoid $A = \{\mathbb{R}^{>0}, \circ\}$ to monoid $B = \{\mathbb{R}^{>0}, \diamond\}$. In answers (a) and (d)—by the way, these are the geometric and harmonic means, quite important concepts on their own—the monoid A is *idempotent*, that is, $a\circ a = a$ for all $a \in A$. But then the image $\Im(\phi)$ of ϕ is also idempotent. Hence $\diamond$ cannot be, for example, the standard addition because $\{\mathbb{R}^{>0}, +\}$ has no idempotents, and if $\diamond$ is the standard multiplication then $\Im(\phi) = \{1\}$ and $\phi(a) = 1$ for all $a \in A$. This is the trivial homomorphism which carries no information about the operation $\circ$.

Answer (c) is not even an operation: it is not defined at $a = b$. It has a singularity there, which makes it even more difficult to represent it by a commutative diagram

of kind of Diagram 1 with a nice, and familiar to children, binary operation $\diamond$ in the arrow at the bottom.

Equation 3 provides a way of constructing an unbelievable variety of commutative diagrams of the kind of Diagram 2. You have to start with an arbitrary binary operation $\diamond$ on $\mathbb{R}^{>0}$ and take an arbitrary 1–1 bijection ϕ on (permutation of) $\mathbb{R}^{>0}$ and define

$$a \circ b = \phi^{-1}(\phi(a) \diamond \phi(b)), \tag{4}$$

then Diagram 2 is obviously commutative. And we are spoilt for choice: there are at least $2^{2^{\aleph_0}}$ permutations ϕ. The only question is

> How many of these commutative diagrams are relevant to school arithmetic and elementary school algebra?

We are afraid, just a handful. Some of them are interesting, but still rather isolated.

For example, we can take $\diamond = +$, and $\phi(x) = x^n$, n a natural number. We get a commutative diagram

(5)

$$\begin{array}{ccc} (a,b) & \xrightarrow{\circ} & \sqrt[n]{a^n+b^n} \\ {\scriptstyle x\mapsto x^n}\,\vdots & & \vdots\,{\scriptstyle x\mapsto x^n} \\ (a^n,b^n) & \xrightarrow{+} & a^n+b^n \end{array}$$

The case $n = 2$ is exhibited by Fletcher as 'Pythagorean map':

(6)

$$\begin{array}{ccc} (3,4) & \xrightarrow{\texttt{Pythag}} & 5 \\ \texttt{Sq}\,\vdots & & \vdots\,\text{Sq} \\ (9,16) & \xrightarrow{+} & 25 \end{array}$$

—but the diagram gives no indication of how the Pythagoras' theorem has been proved.

10. Conclusion

In this brief discussion, it has to be emphasised:

> Questions in the solution are supposed to be formulated by **Learner**, not by a teacher. **Learner has to be in control.** [5]

What we have seen:

- Careful analysis of data.
- A sequence of 'questions' which naturally becomes an algorithm.

Also, concepts from Computer Science:

- Typed variables, aka 'named numbers'.
- Evaluation by call by name and by call by value.
- Reification as a way of introduction of intermediate parameters.

11. Some socio-political comments

L'enfer, c'est les autres.
Jean-Paul Sartre, *Huis Clos*, 1943

The old (1996!) paper by Neil Koblitz[11] *The case against computers in K-13 math education* [20] is well worth re-reading. His warnings are still relevant and fully apply to the possibility of badly prepared, underfunded, hastily introduced computerisation of mathematics education in schools:

> *The downside can be divided into several broad areas:*
> - *drain on resources (money, time, energy);*
> - *bad pedagogy;*
> - *anti-intellectual appeal;*
> - *corruption of educators.*

We hope this paper may serve as a warning: proper introduction of "computational thinking" in school mathematics is an immensely complex and highly expensive task.

We can only applaud the heroic efforts of hundreds of teachers in various countries who are trying to do something in this direction, and we think it is very important to help them in every way available, their experience needs to be studied, systematised, and fed into the development of a potential reform. Very often they are some of the best teachers in their countries. Unfortunately, this usually means that methods and approaches discovered and developed by them are, by default, not scalable and not transferable to the entire country, and for a basic reason: most other teachers are not like them, they are less educated, less motivated, and frequently demoralised by the daily grind of school work.

A serious reform requires a systematic re-education of a whole army of teachers and providing them with (paid, of course) time for personal professional development. An immediate consequence: we will need more teachers.

Lessons of Kolmogorov's reform of school mathematics curriculum in Russia in the 1970s [8] need to be learnt. Andrey Kolmogorov was the world famous mathematician, he had best intentions, but his reform spectacularly failed, and one of the reasons for that was underestimating the role of teachers, even direct neglect of teachers. What is really sad is that Russia at that time had a well developed and functional system of preparing mathematics teachers via a numerous pedagogical colleges (4 years or tertiary education) and mathematics departments in regional universities (5 years)—but this resource had not been engaged. Arguably, this educational powerhouse degenerated over the years, but still the situation in Russia is unlikely to be as bad as that in the UK where, in the words of Tony Gardiner (the best expert in mathematics education in Britain) [16],

> *We know of no other system that pretends to train mathematics teachers by placing small groups of trainees at the mercy of teachers with no relevant ITE [Initial Teachers' Education] experience beyond being themselves teachers, with much of the input being "generic" rather than subject-specific. England appears to be alone among developed nations in embracing such an approach.*

[11] Neil Koblitz is one of the founding fathers of algebraic cryptography, he proposed the elliptic curve cryptography.

Recalling again the tragic fate of Andrei Kolmogorov and his reform [8], we reiterate our warning:

> Any attempt of a deep reform of mainstream mathematics education is a huge task. It is dangerous to undertake it lightly.

Acknowledgements

The first author is forever grateful to Martin Hyland who encouraged him to get involved in this project, and thanks Ian Benson, Tony Gardiner, and Victor Sirotin for many useful discussions and corrections. His thanks go to Rick Booth, Anja Meyer, and Natasa Strabic for their understanding, support and helpful comments.

Disclaimer

The authors write in their personal capacities and the views expressed do not necessarily represent the position of their employers or any other person, corporation, organisation, or institution.

References

[1] T. Altınel, A. Borovik, and G. Cherlin. *Simple Groups of Finite Morley Rank*, Amer. Math. Soc. Monographs Series, Amer. Math. Soc., Providence, RI, 2016,

[2] I. Benson and J. Thorpe. Thinking with arrows for mathematical thought. *Mathematics Teaching*. Association for Teaching of Mathematics, 281 (2022).

[3] M. Boersma. *Domain-Specific Languages Made Easy*. Manning Publication, 2020.

[4] A. V. Borovik, *Mathematics under the Microscope: Notes on Cognitive Aspects of Mathematical Practice*. Amer. Math. Soc., Providence, RI. 2009.

[5] A. Borovik. Being in control. In *Understanding Emotions in Mathematical Thinking and Learning* (U. Xolocotzin (ed.). Academic Press, San Diego, 2017, pp. 77–96. `http://bit.ly/2d6Encg`.

[6] A. V. Borovik. Economy of thought: a neglected principle of mathematics education, in *Simplicity: Ideals of Practice in Mathematics and the Arts* (R. Kossak and Ph. Ording, eds.). Springer, 2017, pp. 241–265. `http://bit.ly/293orpk`.

[7] A. V. Borovik. Mathematics for makers and mathematics for users. In *Humanizing Mathematics and its Philosophy: Essays Celebrating the 90th Birthday of Reuben Hersh* (B. Sriraman ed.), Birkhauser, 2017, pp. 309–327. `http://bit.ly/2qYHtst`.

[8] A. Borovik. The Kolmogorov reform of mathematics education in the USSR. To appear as a chapter in D. De Bock (Ed.), *Modern Mathematics—An International Movement?*, Springer Nature, 2022.

[9] A. Borovik and T. Gardiner. *The Essence of Mathematics – Through Elementary Problems*.Open Book Publishers, 2019. 396 pp.

[10] A. Borovik, Z. Kocsis, and V. Kondratiev. Mathematics and mathematics education in the 21st century, 2021. arXiv:2201.08364 [math.HO].

[11] A. Borovik and Ş. Yalçınkaya. Adjoint representations of black box groups $\mathrm{PSL}_2(\mathbb{F}_q)$. *J. Algebra* 506 (2018) 540–59

[12] A. Borovik. *Shadows of the Truth: Metamathematics of Elementary Mathematics*. In preparation, draft: `www.borovik.net/ST.pdf`.

[13] A.P. Ershov Programming, the second literacy. *Microprocessing and Microprogramming* 8 (1981) 1–9.

[14] T. Fletcher. Thinking with arrows. *Mathematics Teaching*. Association for Teaching of Mathematics, 057 (1971).

[15] A. D. Gardiner. *Teaching Mathematics at Secondary Level*. OpenBook Publishers, 2016.

[16] A. D. Gardiner. Towards an effective national structure for teacher preparation and support in mathematics, *The De Morgan Gazette* 10 no. 1 (2018) 1–10.

[17] S. Gerofsky. A linguistic and narrative view of word problems in mathematics education. *For the Learning of Mathematics* 16 no. 2 (1996), 36–45.
[18] M. Fowler. *Domain Specific Languages*. Addison-Wesley Professional, 2010.
[19] V. Khalin, N. Vavilov, and A. Yurkov. The skies are falling: Mathematics for non-mathematicians. Submitted.
[20] N. Koblitz. The case against computers in K-13 math education (Kindergarten through Calculus). *The Mathematical Intelligencer* 18, no. 1 (1996).
[21] A. Semyonov, S. Polikarpov, and T. Rudchenko. The future of mathematics education. *Mathematics in School* 1 no. 114 (2022).
[22] V. Sirotin. *SI Units*, 2022. `https://github.com/vsirotin/si-units`.
[23] F. Viéte. *The Analytic Art*. (Translated by T. Richard Witmer.) Dover Publications Inc., Mineola, NY, 1983.
[24] M. Voelter et al. *DSL Engineering. Designing, Implementing and Using Domain-Specific Languages*. 2010. `http://dslbook.org`.
[25] О.В. Алексеева и И.Н. Ищенко. *Методика обучения решению текстовых задач в начальной школе*. Амурский гуманитарно-педагогический государственный университет, Комсомольск-на-Амуре, 2019.
[26] И.В. Арнольд, Принципы отбора и составления арифметических задач. Известия АПН РСФСР. Вып. 6 (1946) 8–28. Reprinted in Арнольд И. В. *Принципы отбора и составления арифметических задач*. Moscow, МЦНМО, 2008.
[27] Н.Г. Баженова и И.Г. Одоевцева. *Теория и методика решения текстовых задач*. Флинта, Москва, 2012.
[28] А.В. Белошистая. *Методика обучения математике в начальной школе*. Москва, Владос, 2007.
[29] ЯКласс, Текстовые задач и их решение арифметическим способом. Accessed 11 October 2022.

Email address: `alexandre`$\gg at \ll$`borovik.net`

Email address: `kondratjew239@gmail.com`