Olympiads in Informatics, 2016, Vol. 10, 99–109

DOI: 10.15388/ioi.2016.07

Homo Informaticus – Why Computer Science Fundamentals are an Unavoidable Part of Human Culture and How to Teach Them

Juraj HROMKOVIČ
Department of Computer Science, ETH Zürich
e-mail: juraj.hromkovic@inf.ethz.ch

Abstract. The goal of this article is on one hand to introduce informatics as a scientific discipline in the general context of science and to outline its relationships especially to mathematics and engineering, and on the other hand to propose a way how to integrate computer science into school education.

Keywords: teaching, computer science, mathematics.

1. Introduction

The development of human society is mainly determined by the ability to derive knowledge and to find efficient ways of applying it. Let us explain this claim more carefully. Human beings sample experiences by observations and experiments and use them to generate knowledge by combining their experience with logical thinking. The derived knowledge is used to develop procedures in order to reach concrete goals. A crucial point in our considerations is that to use such procedures one usually does not need to fully understand the knowledge used to obtain such procedures. To an even higher extent, one does not need to understand the way in which this knowledge was derived and verified. Let us illustrate this on a simple example.

The famous theorem of Pythagoras claims $c^2 = a^2 + b^2$ in any right triangle, where c is the length of the longest side (hypotenuse). This theorem was used to fix the right angles when building houses and temples in the classical antiquity. The workers only needed to build a triangle of sizes 5 units, 4 units, and 3 units in order to get a right angle, and for sure they did not need to understand how the theorem of Pythago-

ras was discovered and how it was proven. Hence, to successfully apply the derived knowledge, one did not need to master the high qualification of an investigator. In this way, scientists changed and still change society and became a crucial factor in human development.

Computer science was created as a natural step forward in the above described development, when the following two conditions were satisfied:

1. Using the exact language of mathematics, one was able to discover and describe procedures in such a way that no intellect was needed to apply them. Executing them step by step, everything was unambiguous, and no educated or trained person was needed to find the right interpretation of the particular instructions of the procedure executed.
2. The technology enabling to execute the discovered procedures by machines was developed, and languages providing the opportunity to "explain" universal machines what they have to do were designed.

In this context, we speak about automation. We let the machine execute not only physical work, but also such human work considered as intellectual work in the past. This is the reason why computer science is a mixture of mathematics and engineering.

On the one hand, one uses the concepts, methods, and the language of mathematics in order to understand the considered entities and their relations so exactly that algorithms as unambiguously interpretable procedures can be developed in order to solve a variety of problems everywhere in science, technology, and everyday life. Here, mathematics is the instrument that has to be mastered together with the specific area in which one tries to solve problems.

On the other hand, developing and improving the enabling technology is mostly engineering. This is not only about hardware, but especially about software enabling us to conveniently communicate with computers in high-level programming languages.

We see that computer science is a part of the natural development of science, and it became a discipline that is crucial for the performance of the human society. This contextual view is important when thinking about what computer science actual is, and how to teach computer science in schools. We are asked not to teach specific isolated facts, methods, and other final products of scientific work, but the way how they were discovered, i. e., the paths from the motivation coming from the general context of science to the final products of the research work and engineering. As in its fundamentals computer science is very strongly related to mathematics as its basic research instrument, we start with the question "What is mathematics and how to teach it?" in the next section, and use the derived point of view in the final section to propose the way of teaching computer science in schools. Before the final section, however, we discuss engineering as a missing subject in schools in Section 3. Again, we use this discussion in order to show in the final section how teaching computer science can contribute to understanding some basic concepts of engineering and how it integrates them in the school curriculum.

2. Mathematics as the Language of Science and Consequences of this View for Teaching It

What is mathematics? If you pose this question, you can get a lot of different answers; frequently, even the response that this question is too hard to be answered satisfactorily. A mathematician or a university student can tell you with high probability that she or he proves theorems and thus investigates the structure and the properties of artificial mathematical objects, which can be useful to model reality, or the relationships between different such objects. A high-school student can tell you that mathematics consists of calculations that can be used to solve some classes of mathematical tasks (also called problems) such as solving quadratic equations or systems of linear equations or analyzing the properties of a curve. Obviously, you can also get a response that mathematics is something that is hard to understand, and that every "normal" human being can exist and be successful without it. This is a frequent answer, and unfortunately it is more of a rule than an exception.

Mathematics is the most powerful instrument humans ever have developed in order to investigate the world around us. But it is taught in such a way that the students do not realize this fact. Especially in high schools, they learn methods (algorithms) to solve some given problems (for instance, to find a maximum of a function). This may be also viewed positively as an intellectual challenge, because several methods are not easy to manage. But the bad news is that they can learn to successfully apply methods for solving different problems without really understanding why these methods work properly. High-school students often do not have a good intuition of what infinity or limits are about, but they use these concepts in a fuzzy way in order to analyze artificial functions without seeing any relation to reality. What are we doing wrong? A lot. We have to start to think first about what mathematics is, and then to try to find a way how to teach mathematics in a proper way.

From my point of view, the best way to view mathematics is as a special language developed for science, i. e., for knowledge generation. Going a few thousand years back, people wanted to discover "objective" knowledge. The important word here is the word objective. If you want to reach that, first of all you need a language in which each statement has an unambiguous interpretation for everybody who mastered this language. How to reach this? First of all, you need to give an absolutely exact meaning to the words (notions) you use, because the words are semantically the corner stones of each language. In this context, mathematicians speak about axioms. Many people have the wrong impression that axioms are claims in whose truthfulness we believe, but of which one is not able to prove that they are true. This is mostly not the case. Axioms are the precise definitions of basic notions that describe our intuition about the meaning of these notions. Probably the first concepts people tried to fix were notions such as number, equality, infinity, point, line, distance, etc. What is very important to observe is that people needed hundreds and in some cases thousands of years to come up with such definitions that the community of philosophers and later mathematicians has accepted. Why was all of this done? First of all, the language of mathematics is able to describe

objects, structures, properties, and relationships in an unambiguous way. In this context we speak about the "descriptional" power of mathematics. Thus, people were able to formulate exact claims this way, and so to express our knowledge unambiguously. But this was only one side of the language of mathematics. The language of mathematics was also used to derive new knowledge from the given knowledge, i. e., as a knowledge generator. Leibniz formulated this role of mathematics in a very nice way. He wanted to omit all political discussions and fighting in different committees by simply expressing the real problems in the language of mathematics, and then using calculations and logical derivations to obtain the right solution. Interestingly, he called this "automation" of the human work. By now we know that this dream of Leibniz cannot be achieved. There are two reasons for that. First of all, because of its exactness, the language of mathematics is restricted in its descriptional power, and so we cannot translate all real-world problems into this language. Secondly, one of the most important discoveries of the last century made by Gödel tells us that the argumentational power of the language of mathematics is smaller than its descriptional power. This means that one can formulate claims in current mathematics for which there do not exist proofs of whether they are true or not.

What do scientists learn from that? The development of the language of mathematics is similar to the development of natural languages. You need to create new words and describe their meaning in order to increase its descriptional power, and to be able to speak about things you were not able to speak about before. Moreover, you need new concepts and words to be able to argue about matters you were not able to argue before, i. e., in order to strengthen your capability of deriving new knowledge by thinking. A very important point is that the process of developing the language of mathematics is infinite. As long the number of basic, axiomatic words of the language of mathematics is finite, one can always formulate claims that are not provable within this mathematical language, and one has to introduce new words in order to be able to prove them and so to make new discoveries.

A very nice example is the notion of infinity. Nobody saw anything infinite in the real world, and even most physicists believe that the universe is finite. This means that infinity is something artificial, simply an artificial product of mathematics. But without this concept, most of the current science would not exist. Without the notion of infinity, there would be no concept of limits and so we would not be able to exactly express notions like actual speed or actual acceleration. Our science simply would be somewhere before the discoveries of Newton. Hence, without the "artificial" concept of infinity, one is strongly restricted in the ability to discover our finite, real world.

Another example of a crucial notion is the concept of probability. Most of the sciences, even the non-exact ones such as didactics, psychology, medicine, and economy, heavily use this concept to model and investigate reality, and if they would make predictions without this concept, they would have serious trouble to convince society that their results are trustable to some extent and not only expert opinions.

Why did we focus on the view of mathematics presented above? Because it is the nature of mathematics that shows us which changes are necessary in order to improve

the education in mathematics for everybody. Based on this, we recommend to adopt the following concepts:

1. Focus more on the genesis of the fundamental notions (concepts) of mathematics. To define them took centuries, to prove most of the theorems took a few years. Each new concept enabled to investigate so many things that no single discovery can compete with introducing a fundamental concept. The strengthening of mathematics as a research instrument is the main job of mathematics, and deriving new concepts (not necessarily on the axiomatic level) in mathematics provides the best, true picture of its nature. Without this, nobody can really understand its role and usefulness. And only if one understands the genesis of mathematics as the development of a language of science and as a research instrument, one can be able to apply it regularly to all areas of our life. Teaching mathematics this way can completely change the behaviour of the members of our society. Instead of memorizing and sampling facts provided, one would start to verify the degree of trustability of claims sold as knowledge and to understand to which extent and under which conditions one is allowed to take them seriously.
2. Concrete examples first, abstraction as a final discovery. One first has to touch concrete problem instances and objects in order to get some intuition about their properties. Then, one can formalize her or his intuition into a formal concept. One has to follow the natural way of discovering that usually goes from concrete to abstract. To sell methods and theorems as final products is as poor as teaching manuals for washing machines or Microsoft office instead of teaching discoveries of physics, mechanics, and computer science that enabled to develop these products.
3. Teach algorithmics instead of training calculation methods. Pupils learn in schools to multiply arbitrarily large integers in decimal representations, to solve quadratic equations and systems of linear equations, or to analyze functions, etc. In all cases, most pupils learn to apply given methods, but most of them do not understand why they work. It is more of a challenging memorizing than a deep understanding of the nature of the algorithms used. One has to start to introduce problems instead of presenting methods solving them, and ask the pupils and students to solve concrete problem instances first and finally to discover an algorithm as a robust procedure that is able to solve any of the infinitely many concrete problem instances of the given problem. Discovering algorithms as well-functioning calculation procedures offers another quality of education in mathematics than executing a given calculation method, which any pocket calculator can do faster and more reliable. Teach programming as the art of exactly describing the methods discovered in an unambiguously interpretable way in the language of the machines, and strengthen the ability of exact communication this way.
4. Teach the principles of correct argumentation. Teach the notions of implication and quantifiers, and train direct and indirect proofs. Do not believe that the pupils in high schools cannot learn to verify and to derive simple proofs. They did not manage this in the past, because there was no effort made to teach proving claims, or most effort in this direction was done in a wrong way.

5. Guarantee the opportunity to the pupils to deal with the subjects as many time as needed at an individual speed. Mathematics is one of the sciences that needs a large number of iterative touchings of particular topics until one is allowed to say "Eureka," and gets a reasonable understanding of what it is about. The trouble is that no teacher can assure this by herself or himself for everybody in the class. Another problem is that most textbooks of mathematics are good collections of exercises, but explanations are written more for teachers than for pupils. One way out is to change the style of the textbooks. The textbooks should be written in such a way that pupils and students would be able to learn from them by their own with a minimal support from outside. Partitioning the discoveries into a number of small, natural steps written in the language mastered by the pupils at the corresponding age, and regularly giving the opportunity to verify whether one understands the topic up to now properly are some of the basic principles used to create good textbooks for mathematics.

Finally, one can ask how to reach the new teaching style for mathematics described above. For sure, one cannot ask the high-school teachers to make this change without showing them how to do this in detail. One also cannot ask educationalists, who do not have a sufficiently deep contextual knowledge of mathematics to master these changes. The movement has to start at the universities, where the teaching style has to change first. In order to speed up this process in Switzerland, in our department at ETH Zürich we develop new textbooks for teaching different topics of mathematics and computer science for all school levels. Our experiments prove that mathematics can become one of the most favorite subjects of pupils and students if taught in the way described above. Students can successfully master topics that were considered to be too hard for them before, and the marks in mathematics can be significantly higher than the average marks over all topics.

For me, it is not a question of whether the proposed evolution of the education in mathematics will come. It is only the question of the time at which particular countries will need to adopt it. Since this is a service for the future generation, the earlier the better.

3. Why Engineering is Not Allowed Not to Be a Part of Basic Education

As mentioned in the introduction, human society uses the knowledge discovered in order to reach different goals more efficiently by developing various procedures or different products. This is highly creative work that is beyond the pure learning of facts and making calculations that can be completely automatized. The whole process of engineering work starts with a description of the goals to be achieved. After that, one starts to combine experience and fundamental knowledge of science in order to design a solution that has to be implemented as a prototype. Next, one has to test this prototype, modify, and improve it until an acceptable product is produced.

In today's schools, we find almost nothing about the concept of iterative specifying, testing, modifying, and improving the product of our work, let alone about fundamental constructive ways of creating original products. But this is fundamental to human activities since the beginning of time. The current school systems ignore this fact to a high extent and are more about teaching to memorize than teaching to work in a creative way. One can explain this educational misconception by the fact that, in contrast to basic scientific models of reality, the engineering work is heavily dependent on experience that is often hard to formalize and thus to teach. The creative work of engineers as human experts cannot be described by an algorithm (a method). However, this must not be a reason to remove engineering from education, because students also have to learn to build their own experience over a longer period of time in order to become an expert for a special area.

The crucial fact we want to point out is that computer science mastered to formalize several basic concepts of engineering and made them available to our schools as a result. Teaching computer science is a chance to introduce engineering as a highly creative, constructive activity to our educational system.

Let us consider some concrete illustrations. Probably the most fundamental concept of technical sciences is modularity. One builds some simple units with clear and well verifiable functionalities and calls them modules. Then one uses these modules as fundamental building blocks to construct more complex systems that are again considered basic modules for constructing even more complex systems. There is no upper bound on the growth of the complexity of systems built in such a way. Unbelievably many human activities can be described by this modular concept. This includes learning. Ones learns some simple facts and methods and their applications in a restricted framework. After that, one masters them perfectly in such a way that one can use them as elementary operations in attacking more complex tasks. All spiral curricula run in a well-designed modular way. Thinking in a modular way when trying to reach given goals is the most fundamental instrument of creative human work. And it does not matter whether you apply it top-down or bottom-up. If one applies a top-down approach, then everything is clear and one designs only a transparent and well-verifiable plan for constructive work. More typical is the bottom-up approach where one builds more and more complex modules without knowing what the final product will be about.

Programming is an excellent instrument to teach modularity. Writing programs to automate solving different tasks provides modules for attacking much more complex tasks. Children of age between 10 and 12 are able to put five loops into each other without being disturbed by or even observing the complexity of their final product due to modularity. One simple program containing a loop gets a name and so becomes a new instruction. This instruction is used in the body of another loop that also becomes a new instruction used in another loop, etc. Here, one can train both, the top-down approach as well as bottom-up approaches devoted to open-end tasks. To practice the ability to bring a clear structure into complex processes is at least as important as any of the fundamental concepts of sciences or humanities contained in the school curricula.

Another important concept is teaching to test or verify products of our work. Typically, children verify their own products by asking teachers or adults to tell them whether their work resulted in what they were asked to reach. But this way, they cannot sufficiently build their self-confidence because they need a strong dependency on their supervisor. They have to learn to trust their solutions by being taught to verify the products of their work by themselves. Programming is again a wonderful instrument for this purpose using the platform or editor that fits the educational requirements. One can learn to test the functionality of programs by running them on several data sets as well as by the verification approach based on a transparent structuring of the program into small modules whose correct behavior can by easily verified.

In general, constructive engineering work within the educational system changes the behavior of young people in an essential way. They move from the role of customers searching for products with some desired property to the role of creative inventors for designing and developing products with new, original functionalities. The never-ending story of improving any human products with no limits on what can be approached in the future would be the most important, great consequence of embedding the creative way of thinking of engineers into school systems.

4. Teaching Computer Science as a Fundamental Step in the Evolution of Our Educational Systems

In the two previous sections, we already outlined, with respect to improving teaching of mathematics and introducing engineering, the principal contributions that could be offered by teaching computer science in schools in a proper way. This word "proper" is crucial for us, and thus we start by listing what we are not allowed to do if we want to avoid a disaster when introducing computer science to schools. In what follows, we present the most frequent mistakes that already caused frustrations in different countries.

1. To teach how to work with concrete software products and call it computer science. This activity destroyed the image of computer science as a scientific discipline in the past.
2. To let computer science be taught as a part of "social media" by teachers educated in human sciences only, and focusing on social, emotional and psychological aspects of communication by new technologies.
3. To choose the topic to be taught by committees of experts offering their favorite topic without looking at the whole context of science as presented above in Section 2.
4. To sell computer science as the ability to work with computers.
5. To sell computer science as a special branch of mathematics.
6. To sell computer science as a pure engineering discipline.
7. To try to teach the newest achievements of computer science. Think about what would happen if physics would try to do that instead of following the history, and thus developing step by step our view of the physical world.

8. To try to sell computer science as a joy much easier than mathematics and physics by avoiding any depth and thus a spiral curriculum, and instead presenting one simple application after the other.
9. Going too much into technical details about concrete programming languages, software systems, or hardware.

While 1, 2, and 4 have been the main reasons for destroying the image of computer science in the society in the past, currently the points 7 and 8 are the major danger for establishing computer science as a school subject.

After listing what we are not allowed to do, it is now time to switch to positive recommendations and conceptual work. We do not want to make a proposal for the content of a computer science curriculum, because our goal is not to go too much into detail, and because, for sure, there are various possibilities for good implementations of the computer science subject in schools. What we try here is to recommend a strategy and principles that are useful for designing a computer science curriculum that can be accepted as a fundamental part of education in its generality for everybody, and that essentially contributes to:

1. The understanding of our world (in this case with the focus on the artificial world created by humans).
2. Developing our way of thinking in a dimension that cannot be compensated by teaching other school subjects.
3. Providing knowledge that is useful and sometimes even expected as prior knowledge for the study of a variety of specialized scientific disciplines later (university studies, etc.).

As a byproduct, we have to aim to improve teaching overall, especially by strengthening the subjects mathematics and natural sciences. We already presented the basic strategy how to design a computer science curriculum in Section 2 about mathematics. We have to follow the genesis of computer science and think about motivations and fundamental concepts introduced and discovered by computer scientists from the point of view of science as a whole. For sure, we have to think about or even discover which concepts are available to which extent in which age, and to follow all the ideas for creating good teaching materials as presented in Section 2. Let us be more concrete and present a few examples.

One can decide to introduce programming at the age between 8 and 14. The first step is to deal with abstractions that enable us to unambiguously describe the problem instances. Then we teach to sample experience by trying to find solutions to concrete, special problem instances, whose size and complexity may grow with growing experience. After some time, one can develop a strategy that works successfully for a small collection of problem instances that we subsequently call a problem. Having a solution strategy, one has to learn to communicate it, i. e., to unambiguously describe it for anybody else. After that, we are allowed to start teaching proper programming by describing our strategy as a program in a suitable programming language. We are not allowed to teach the list of all instructions (fundamental words) of a programming language. We have to start with very few (10 to 15) fundamental instructions, and use modularity to

create new words (instructions) in order to make our communication with the computer more convenient. After writing programs, we let them run in order to verify their correct functionality and learn to correct, improve, and modify our programs in order to get a final product with which we are satisfied. Let us list some of the added values when teaching the introduction to programming in the way described above:

1. Training and strengthening the abstraction in representing real situations by drawing graphs, writing lists or tables with different kinds of elements.
2. Contributing to teaching mathematics by searching for a solving strategy, instead of simply learning a method as a given product of the work of others.
3. Strengthening the ability to express matters and procedures in an exact way, and so to improve the way used to communicate.
4. Recognizing that a language is not a given final product of human work, but that any language is continuously developed, and that in case of a programming language one can develop the language on her or his own with respect to her or his personal demand.
5. Defining new instructions by describing the meaning of the new words by subprograms, one learns the principle of a modular design that is common and fundamental in engineering.
6. Introducing the concepts of testing, verifying, modifying, and improving is the first contact with the creative, constructive work of engineers.

A really good teaching sequence for introductory programming can be created if one focuses on the above listed added values and not on technical details of programming languages and other software used or on a specific class of tasks.

Another nice example is teaching cryptography. Cryptography can be viewed as the history of developing the notion "secure cryptosystem." One can start with the historical examples in order to introduce the basic terms decryption, encryption, key, and cryptosystem with a lot of creative work by designing and breaking new, own cryptosystems. After defining the concept of "security" by Kerckhoff, one can build the bridge to probability theory. The concept of probability was used to design new cryptosystems and later to break them. One can wonderfully understand the importance and the usefulness of the concept of probability studying the history of secret communication in this way. Then one can introduce the formal mathematical definition of absolutely secure cryptosystems with respect to the concept of probability, and recognize that such system cannot be built for practical purposes. Finally, the concept of computational complexity offering public-key cryptosystems is the way out, leading to the recent e-commerce.

What we try to repeatedly present as the key strategy is to follow the history of the discoveries of particular concepts, methods, and ideas, and not to try to sell finalized products of science. The creative work is the most (and may be even the only really) exciting part of the study. Let us teach creativity by repeatedly discovering things that were already discovered, up to the point where one is able to discover something completely new. Forget about teaching facts, teach how to verify the trustability of claims made by others. We are lucky, because we are allowed to create a curriculum for a

completely new subject. We can implement principles, which the other subjects still did not recognize, and so contribute to the evolution of the system of education. For those who would like to see detailed implementations of the design principles presented above, we recommend to following textbooks from our production (Böckenhauer and Hromkovič, 2013; Freiermuth *et al.*, 2014; Hromkovič, 2011; Hromkovič, 2014) or the book "Algorithmic Adventures – from Knowledge to Magic" (Hromkovič, 2008; Hromkovič, 2009).

References

Böckenhauer, H.-J., Hromkovič, J. (2013). *Formale Sprachen.* Springer Vieweg.
Freiermuth, K., Hromkovič, J., Keller, L., Steffen, B. (2014). *Einführung in die Kryptologie.* 2nd Edition. Springer Vieweg.
Hromkovič, J. (2011). *Berechenbarkeit.* Vieweg+Teubner.
Hromkovič, J. (2008). *Sieben Wunder der Informatik – Eine Reise an die Grenze des Machbaren.* Vieweg+Teubner.
Hromkovič, J. (2009). *Algorithmic Adventures – From Knowledge to Magic.* Springer.
Hromkovič, J. (2014). *Einführung in die Programmierung mit LOGO.* 3rd Edition. Springer Vieweg.

J. Hromkovič is professor of informatics with a special added focus on computer science education at ETH Zurich. He is author of about 15 books published in 6 languages (English, German, Russian, Spanish, Japanese, and Slovak) and about 200 research articles. He is member of Academia Europaea and the Slovak Academic Society.