Computational Thinking: Possibilities and Challenges

Leila Ribeiro*, Daltro José Nunes *, Marcia Kniphoff da Cruz † and Ecivaldo de Souza Matos ‡
* *Institute of Informatics, Federal University of Rio Grande do Sul (UFRGS),*
Porto Alegre, Brazil
Email: {leila,daltro}@inf.ufrgs.br
† Departamento de Informática, Universidade de Santa Cruz do Sul(UNISC),
Santa Cruz do Sul, Brazil
Email: marciakniphoff@gmail.com
‡ *Instituto Federal de Educação, Ciência e Tecnologia de São Paulo*
São Paulo, Brazil
Email: ecivaldomatos@yahoo.com.br

***Abstract*—The aim of this paper is to discuss the importance of computational thinking and how to include techniques to teach this kind of ability in schools in Brazil. First, we discuss what Computational Thinking is and the importance of teaching such skill in school. Then, we list some of the challenges that are involved in introducing this discipline in school curricula.**

***Keywords*-computational thinking, elementary and middle school, high school, education in Brazil**

I. Introduction

Computer Science can be seen as *the art of solving problems*, in the sense that in Computer Science one studies how to construct and organise the solution of a problem, presenting the process that leads to the solution in a very precise way. This process is called *algorithm*, and it can be described at various levels of abstraction. Students learn in elementary school ways to sum, divide and multiply numbers, extract the square root, etc. All these are examples of algorithms, that is, there is a series of well-defined steps that they follow to perform each of these operations to get the from the arguments to the result. Algorithms may be used in any area, not just to describe mathematical operations. For example, we may use algorithms to describe how a cake is made, how to change a tire, how to perform a biological experiment, how to organise a party, etc. A very important characteristic of an algorithm is that it describes precisely and non-ambiguously the process that has to be executed such that different persons (or even machines) following the algorithm can perform the same steps in the same way, arriving to the same results (when the process is deterministic).

The question of which kind of procedures can be described in such a precise way actually led to the development of the area of Computer Science. In the kernel of this area lies the fundamental work of Alan Turing [1] and other researchers like Church, Gödel, Kleene, ..., explaining what is an algorithm, that is, how can we define the process of solving a process in a precise way. Turing's idea led to the development not only of techniques to solve and analyse the complexity of problems, but also to the architecture of the first machines that could automatically follow a series of steps to solve a problem, called *computers* nowadays. Since the the dawn of Computer Science many methods to solve problems and describe the solutions in a organised and efficient ways were developed. And, in fact, if we compare problems that could be solved and automatised in computers fifty years ago to the ones that can be handled today, we will see that the complexity and size of problems solved today was non-imaginable 50 or 60 years ago. As we learned how to solve bigger and more complex problems, many different abstractions and techniques were developed. and we claim that this kind of problem solving skills are needed by everybody not just Computer scientists.

In this paper we will first explain what *Computational Thinking* is and how it compares to other abilities, specially to logical and mathematical reasoning. Then we will motivate why this kind of skill is important and finally discuss how to teach this kind of ability at school, pointing out some challenges involved in this process.

II. What is Computational Thinking?

The cognitive process used by human being to find algorithms to solve problems is called *Computational Thinking* or *Algorithmic Thinking* [2], [3]. This process which is the basis of Computer Science, may be applied in many other areas like Mathematics, Physics, Chemistry, Sociology, Philosophy, etc, enabling the students to specify and organize the solution of problems. The study of algorithms involve concepts like abstraction, refinement, modularisation, recursion, etc. Learning these concepts improves the capacity of reasoning and problem solving by way of meta-cognitive learning processes, considered essential for intelligence [4].

978-1-4799-3057-9/13 $31.00 © 2013 IEEE
DOI 10.1109/WEIT.2013.32

To illustrate this idea, we may think of a computational scenario consisting of a person A, with his language L_A, a person B, with his language L_B, and a machine M (a computer, for example) with its language L_M. A receives a problem P to solve (e.g., to extract the square root of a number). After analysing the problem, A chooses a machine, say M, in which the solution can be suitably implemented. The solution is then written in a language L_M understood by machine M (arrow from A to M, in Figure 1). The solution of the problem is known as algorithm. The process of solving problems (involving analyzing problems, selecting adequate machines, constructing and implementing algorithms) is called Computational Thinking.

But note that this approach is not only applicable to mathematical problems, problems can be of any nature, for instance in the context of physics, mathematics, management, law, etc. Now, imagine that A receives another problem P', for example, which is the shortest path that a taxi driver (B) must follow, leaving from address x and arriving at address y? A solution (algorithm) can be: first straight ahead, then turn right, etc. To arrive to this solution, A must first analyse the problem finding out whether there is no solution, or there is only one solution or or there are more than one solutions (in which case, A must choose which one should be adopted). After this, A must describe the solution (algorithm) in the language understood by B (in this example, this is the "machine" that will automate the solution).

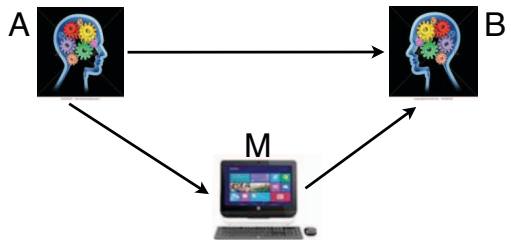

Figure 1.

Computational thinking is different from mathematical and logical reasoning. In Mathematics, a fundamental notion is the concept of number as an abstract for quantity. It also studies how to build more complex objects out of simple ones (e.g. sets, relations, rings, graphs, ...) and how to describe space in an abstract way (geometry, trigonometry, etc.). Change is modelled essentially by using functions and it is possible to describe and analyse complex systems by using may different mathematical tools (differential equations, dynamic systems, chaos theory, etc.). Children learn at school since the early years notions of numbers and how to perform different operations on numbers. These operations are actually concrete algorithms, that is, they learn how to follow a given algorithm to be used on a given abstraction (number). For example, when give two different numbers and asked for its sum, they may choose among the ways they learned to solve this problem, including using a calculator to obtain the result. But, if a student is faced with a different task for which no algorithm was taught, like sort a pile with 20 exams in ascending order of grades, how can we help he/she to obtain the result? With 20 exams, maybe the student will try to sort them without using any technique, just his basic knowledge of numbers and comparison. However, is there are 200 or 2000 exams to sort and the time is limited, which skills will help the student solve the problem, or at least estimate the time that would be necessary to solve it? How can the student know whether the task, or at least part of it, can be done by a computer? Or whether the task could be done faster is he had some help of other students? And in case there is help, how can the task be divided into the students, and the results combined at the end? At this point, the difference between mathematical reasoning and computational thinking may be clear: to sum numbers we just need to know a given abstraction and given algorithms; to sort the pile of exams we need to choose the abstraction, find an algorithm (actually there are many sorting algorithms) that is well-suited to the application, or build one, execute the algorithm (maybe some parts in a computer).

Logics can be seen as a brach of Mathematics modelling various ways in which we can deduce new things from already known ones. The skill of logical deduction is undoubtedly a very important one nowadays. Whenever we have to support a position, it is very convenient to find a deduction sequence that, from well-known and accepted facts, leads to the desired conclusion. An algorithm has similarities with a logical deduction in the sense that we also have an input (the data that will be used by the algorithm / the facts that will be used for the deduction), an output (the result of the algorithm / the conclusion of the deduction) and each step leading from the input to the output must be precisely defined. Theoretically, these processes (building an algorithm / making a deduction) may be equivalent. But in practice there is a fundamental difference: the process of building an algorithm is constructive and we need abstractions and techniques that are different from the ones used in Logics.

Computational thinking is thus the skill that allows us to solve problems in a systematic way, enabling us also to analyse our solutions and execute them using various resources (that may be humans, machines, etc.). To obtain this skill, we need:

- *abstractions* to model reality. Some may be the ones we are used to in Mathematics or Physics, but other kinds

of structures that are related to describing processes are also needed;
- *techniques to master complexity*, like divide and conquer, composition, recursion;
- *techniques to transform* one problem into another, to be able to adapt solutions of already solved problems to new ones;
- *to see data as programs and programs as data*. This is not obvious today for persons outside the Computer Science community, and is essential if we want to understand and teach a theory about building solutions to problems;
- *to understand concurrency* and how to deal with multiple tasks being performed simultaneously to obtain the solution of a problem;
- *to understand different models of computation*, to be able to choose the more appropriate to use for each problem. Models of computation offer radically different views on how processes are described and therefore a solution to the same problem may be very simple if one model is chosen, or very complex if another is chosen;
- *to know standard solution to recurrent problems*, like sorting, to be able to recognise when a problem (or part of it) has already been solved, to compare and choose amongst solutions and to adapt the chosen solution to the new problem;
- *to understand which parts of a solution can be carried out by computers*;
- *to understand the limits of computing and of computers*. The limits of computing tell us which problems can not be solved algorithmically and whose solution can not be completely executed by a machine. The limits of computers are related to concrete machines that can be used to execute algorithms, they tell us whether a solution, although possible, is feasible, in the sense that the solution uses a bounded amount of resources (space, time, machines, ...)

Some of these topics are part of kernel disciplines in a Computer Science course (Algorithms, Computability Theory, Complexity Theory), some are taught in Software Engineering courses.

III. How to Teach Computational Thinking?

Computational Thinking is thus a skill like reading, writing, speaking and performing arithmetical operations. This new skill builds on top of these four, and provides a more sophisticated view of how to approach problems. However, it requires a higher capacity of abstraction. Like other skills, this one can be learned and trained since childhood. Note that the idea is not to teach Computer Science or programming at school. The suggestion is that we go much deeper in teaching the fundaments of Computing Science. To teach many of these concepts, we do not even need computers. For many examples of how to do this, see [5].

Now we will discuss some of the challenges related to teaching computational thinking.

A. Challenge 1: Curricular Changes

The first step to teach Computational Thinking at school is to understand how and when this should be done. Should it be a mandatory discipline, like Mathematics? Which concepts shall be introduced at which levels? In the US there has been a lot of research and workgroup working on this topic for many years, some of the results are consolidated in the Computer Science Teachers Association (CSTA) webpage [6]. But there is still much to be done in clearing out this issue and elaborating a concrete curricula for this discipline in schools.

When we think about teaching of any content is necessary contextualize it within a curriculum design. To introduce Computational Thinking in schools it is necessary deal with curricular changes. The interdisciplinary perspective of curriculum integration [7], [8] with other disciplines of school curriculum is a possibility, but it has other challenges like Computer Science teacher training and political aspects. In this perspective, it is possible to promote meaningful learning through a dialectical movement of assimilation, reflection and internalization, with use of previous knowledge to promote learning. Curricular changes are ideological challenges because the teacher chooses within his philosophical education perspective what, how, when and where he/she will teach.

To follow the guidelines presented in [6], there should be an analysis on how to adapt them to the curricula we have in schools in Brazil. It is possible that each school implement into their formal curriculum latent disciplines that exceed the required curriculum, inserting a discipline that works with Computational Thinking. There are indeed many schools in Brazil that have a discipline on Computing; however, they typically cover only the digital resources, and not Computational Thinking. Without clear guidelines on what should be taught and when, it is very difficult that Computational Thinking will be successfully included in the curriculum of most schools in Brazil.

B. Challenge 2: Building CS Educators

In Brazil and in the rest of the world there is a great deficit in Computer Scientists. The situation is even more dramatic if we consider the number of teachers with solid knowledge in Computer Science that will be needed to introduce Computational Thinking at Elementary and High Schools. According to [9] there are currently in Brazil 110 undergraduate courses on Teaching Computer Sciecne. Despite this number, the challenge of building CS Educators is really huge, since most of these do not focus on the topics listed in the previous section, but rather on teaching how to

use current technologies to build educational software or how to teach students how to use computers. And this is far from teaching Computational Thinking. These professional are not getting the background that is required to teach problem solving skills.

C. Challenge 3: Government Polices

Introducing Computational Thinking at school is very important to form students with highly advanced problem solving skills. This is very difficult to implement and thus has a high cost because we have to invest in forming Educators and re-structuring curricula. To implement such change investments will be necessary in

1) Research in Computational Thinking and how to teach it,
2) Courses to form Educators that are fluent in this area.

The success of a Country depends on its most precious resource: People. Therefore we should to invest whatever is necessary in Education to prepare the citizens that will rule and carry the country in the future. And it is basically the role of the Government to lead this process, not only by financing lines of research and courses as cited above, bur also defining policies to change the curricular configuration of Basic Education and the curricular guidelines for undergraduate courses in which teachers are trained. The Government is also responsible for producing educational materials for schools. This challenge (political) triggers reflection on why some disciplines have spent the past 3 years to be part of the required curriculum, for example in High School, Sociology and Philosophy. Aren't advanced problem solving skills (given by Computational Thinking) not extremely needed by our society? Can any area of knowledge or productive sector dispense the computational means and continue sustainable? It is evident that we need to introduce Computational Thinking in Education and the Government should urgently build workgroups to lead this important task in Brazil. This workgroups must consist not only of professionals of the area of Education, but it is fundamental that experts in Computer Science, specially Theoretical Computer Science, be part (or even lead) this enterprise. Mastering this skill is a very new concept, nobody learned it at school yet, and the professionals that are closer to teaching this kind of subjects are the ones that teach and do research in Computability Theory, Algorithms and Complexity and Logics. Thus, only a joint work between professionals of these two areas (Theoretical Computer Science and Education) will result in the successfully introduction of Computational Thinking at school.

D. Challenge 4: Breaking Established Rules' Challenge

Finally, it is very hard to change established rules and tradition. It is specially difficult to convince people that a knowledge that they do not have, or did no learn at school, is or will be essential in the future. But the speed in which mankind is progressing is enormous, the size and complexity of the problems we are facing now were not even imaginable some years ago, and this is certainly true also for future problems. To cope with these forthcoming problems will require new skills, one of the is Computational Thinking, and we have to provide this skill to the persons that will handle these problems: our students of today.

IV. Conclusion

In this paper we briefly discussed what Computational Thinking is, trying to motivate the possibilities that can be obtained if this discipline is included in school curricula. We also enumerated some challenges that are involved in the task of teaching Computational Thinking at schools, specially in the Brazilian context. These discussions were part of a panel that occurred in the *WEIT 2013 - Workshop-Escola de Informática Teórica*, in October 2013.

References

[1] A. Turing, "On computable numbers, with an application to the entscheidungsproblem," *Proc. London Mathematical Society*, vol. 42, pp. 230–265, 1937.

[2] J. M. Wing, "Computational thinking," *Commun. ACM*, vol. 49, no. 3, pp. 33–35, 2006.

[3] ——, "Computational thinking and thinking about computing," in *IPDPS*. IEEE, 2008, p. 1.

[4] R. Sternberg and J. Mio, *Cognitive Psychology*. Cengage Learning/Wadsworth, 2009. [Online]. Available: http://books.google.com.br/books?id=la6mSK9vyIgC

[5] T. Bell, M. R. Fellows, and I. Witten, *Computer Science Unplugged : offline activities and games for all ages: Teacher Edition*. Computer Science Unplugged, 1996. [Online]. Available: http://csunplugged.org

[6] [Online]. Available: http://csta.acm.org/Curriculum/sub/CompThinking.html

[7] N. J. Gehrke, "A look at curriculum integration from the bridge," *Curriculum Journal*, vol. 9, no. 2, pp. 247–260, 1998.

[8] J. VanTassel and S. Wood, "The integrated curriculum model (icm," *Learning and Individual Differences*, vol. 20, no. 4, pp. 345–357, 2010.

[9] C. Castro and G. Vilarim, "Ca licenciatura em computação no cenário nacional: embates, institucionalizaçã o e o nascimento de um novo curso," *Revista Estudos Academicos - UEM*, vol. 148, p. 18, 2013, in portuguese. [Online]. Available: http://periodicos.uem.br/ojs/index.php/EspacoAcademico/article/view/21635