DOI:10.1145/3231587

Enrico Nardelli

Viewpoint
Do We Really Need Computational Thinking?

Considering the expression "computational thinking" as an entry point to understand why the fundamental contribution of computing to science is the shift from solving problems to having problems solved.

I CONFESS UPFRONT, the title of this Viewpoint is meant to attract readers' attention. As a computer scientist, I am convinced we need the *concept* of computational thinking, interpreted as "being able to think like a computer scientist and being able to apply this competence to every field of human endeavor."

The focus of this Viewpoint is to discuss to what extent we need the expression "computational thinking" (CT). The term was already known through the work of Seymour Papert,[13] many computational scientists,[5] and a recent paper[15] clarifies both its historical development and intellectual roots. After the widely cited *Communications* Viewpoint by Jeannette Wing,[19] and thanks to her role at NSF,[6] an extensive discussion opened with hundreds of subsequent papers dissecting the expression. There is not yet a commonly agreed definition of CT—what I consider in this Viewpoint is whether we really need a definition and for which goal.

To anticipate the conclusion, we probably need the expression as an instrument, as a shorthand reference to a well-structured concept, but it might be dangerous to insist too much on it and to try to precisely characterize it. It should serve just as a brief explanation of why computer science (or informatics, or computing: I will use these terms interchangeably) is a novel and independent scientific subject and to argue for the need of teaching informatics in schools.

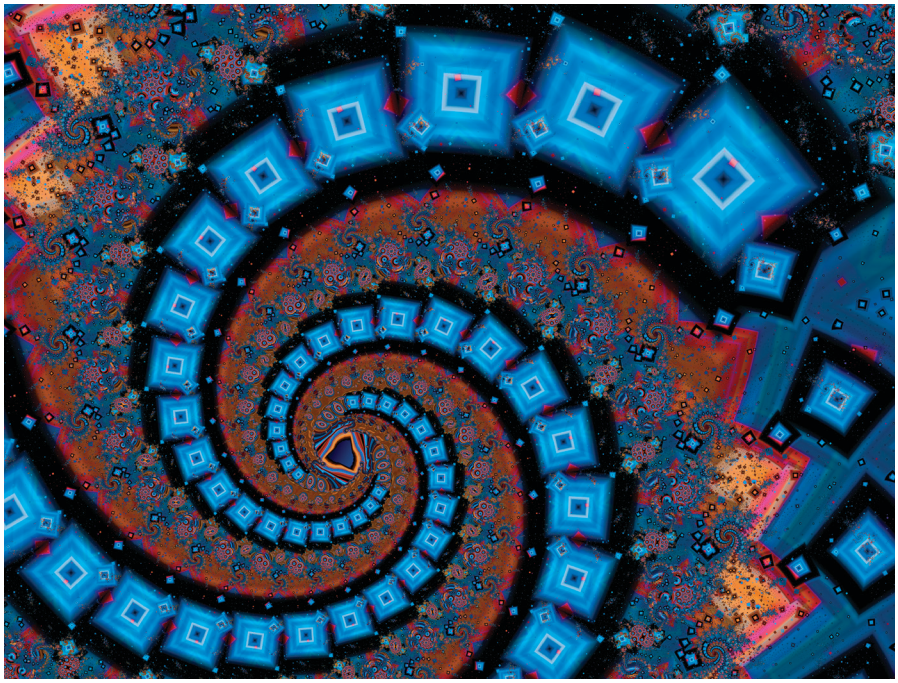


IMAGE BY VALLIA

Wing discussed CT to argue it is important every student is taught "*how a computer scientist thinks*,"[19] which I interpret to mean it is important to teach computer science to every student. From this perspective, what is important is stressing the educational value of informatics for all students—Wing was in line with what other well-known scientists had said earlier; I mention several here.

Donald Knuth, well known by mathematicians and computer scientists, in 1974 wrote: "Actually, a person does not really understand something until he can teach it to a computer."[10] George Forsythe, a former ACM president and one of the founding fathers of computer science education in academia, in 1968 wrote: "The most valuable acquisition in a scientific or technical education are the general-purpose mental tools which remain serviceable for a lifetime. I rate natural language and mathematics as the most important of these tools, and computer science as a third."[9] Even if both citations are not relative to a school education context, in my view they clearly support the importance of teaching computer science in schools to all students.

However, the wide popularity gained by CT after Wing's *Communications*

Viewpoint risks spoiling the original aim. Increasingly, people are considering CT a new subject, somehow different or distinct from computer science. In the quest to identify the definition that Wing did not provide, people are stressing one or other aspect (abstraction, recursivity, problem solving, ...) and in doing so they obscure its meaning. See Armoni[2] and Denning[5] for clear and illuminating discussions of this issue.

This situation becomes even more garbled when it comes to education. Speaking about teaching CT is a very risky attitude: philosophers, rightly, ask what we mean by "teaching thinking"; mathematicians appropriately observe that many characteristics of CT (such as abstraction, recursivity, problem solving, ...) are also proper of mathematics (which they do not call "mathematical thinking"); pedagogues ask how we can be sure CT is really effective in education; teachers want to know which are the methods and the tools for teaching this new discipline and how they can learn to teach it; and parents are alternately happy because it appears school has finally started to align itself to the digital society while they are also concerned about what will happen to their children in the future if they just learn to code with the language of today.

I think a large part of the community of computing scientists and educators is convinced the original *Communications* Viewpoint by Wing was aiming at "start rolling the ball" and what needs to be done is teaching informatics in schools, possibly beginning at an early age. Moreover, I am convinced the same people are fully able to understand the meaning of Wing's expression "to think like a computer scientist" without the need of exactly explaining it. Or, if it is absolutely needed, they might agree with the self-referential sentence "CT is the set of mental and cognitive competences obtained by the study and practice of computer science": the "tacit knowledge" defined by Polanyi.[14]

Already in 1974 Knuth warned, in discussing computer science, that "the underlying concepts are much more important than the name."[10] It is much more so, I think, for CT. What really counts is the fact that computing is taught early in schools. This is actually the path being followed by some major countries. Here, I discuss the three most relevant ones.

In England, the national computing programmes of study,[a] published by the Department of Education in September 2013 and mandatory since school year 2014–2015, uses CT in the presented sense of what one gets by the study and practice of computing. In fact, it uses it in the opening statement "A high-quality computing education equips pupils to use CT and creativity to understand and change the world" and then just two more times, in goals for Key Stage 3 "understand several key algorithms that reflect CT" and KS4 "develop and apply their analytic, problem-solving, design, and CT skills." The curriculum never defines the term.

In the U.S., the "Every Student Succeeds Act" (ESSA), approved by Congress in 2015 with bipartisan support, has introduced computer science among the "well rounded educational subjects" that needs to be taught in schools "with the purpose of providing all students access to an enriched curriculum and educational experience," and does not contain at all the term "computational thinking." In January 2016, President Obama launched the initiative "CS For All" whose goal is "to empower all American students from kindergarten through high school to learn computer science and be equipped with the CT skills they need ...". Once again, CT is what you get when you have learned computer science.

In France, the Académie des Sciences—the highest institution representing French scientists—published in May 2013 the report "L'enseignement de l'informatique en France. Il est urgent de ne plus attendre," ("Teaching computer science in France. Tomorrow can't wait.") recommending—for what regard the teaching of computer science ("informatique")—"teaching should start at the primary level, through exposure to the notions of computer science and algorithms, ... <and> should be further developed in middle and secondary school." Analyzing their use of CT ("pensée informatique"), it is clear that in their vision the term denotes the specific habits of thinking developed by learning computer science. Just a couple of examples: "computing ... leads to a different way of thinking, called CT" and "learning about programming is a way to discover the rudiments of CT."

It emerges, from these three examples, that CT is not a new subject to teach and what should be taught in school is informatics.

But on the other side, the high number of papers published with CT in their title or abstract (the ACM Digital Library alone contains more than 400) indicates a lot of people seem to argue (and even Wing seemed to agree[21]) that CT is something new and different. Some even say "coding" (which they consider different from "programming") is all you need to learn it! A discussion of risks related to this approach and other delicate issues regarding CT appeared in a recent *Communications* column.[8]

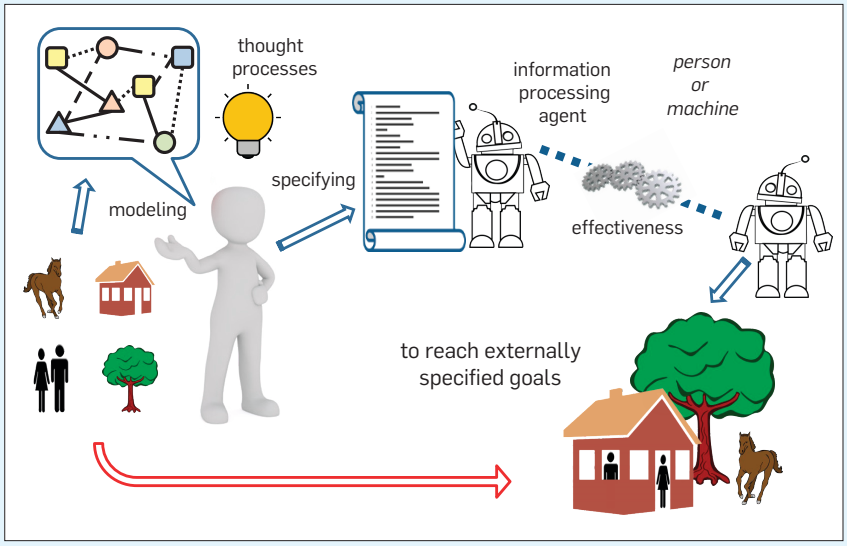
I am convinced that considering CT as something new and different is misleading: in the long run it will do more harm than benefit to informatics. After all, they do not teach "linguistic thinking" or "mathematical thinking" in schools and they do not have "body of knowledge" or "assessment methods" for these subjects. They just teach (and assess competences in) "English"[b] and "Mathematics." Subsequently, the various linguistic (resp. mathematical) competences gained by study of English (resp. Mathematics), beyond being used in themselves, find additional uses in other disciplines. Between CT and computing there exists the same relation. Therefore, we should discuss what to teach and how to evaluate competences regarding informatics in primary/middle/secondary schools, and forget about teaching and evaluating competences in CT.

In summary, speaking about CT helps people understand that: we are focusing on scientific and cultural aspects of computing; *we are not dealing with system and tools, but with principles and methods*; we are focusing on the core scientific concepts of computing, on its *conceptual kernel*.[11] Different from what happens with language and math, we are forced to explicit this distinction since computers are what embodies informatics for most of people. In addition, we do not think the "computer scientists' way of thinking" is better than others, just that it offers a complemen-

a See https://bit.ly/1f7PIFU

b Or the relevant native language.

Modeling a situation and specifying the ways an information-processing agent can effectively operate within it to reach an externally specified (set of) goal(s).



tary and useful conceptual paradigm to describe reality.[7]

At this point people usually ask which is this "conceptual kernel" and which examples can we provide. This is a critical passage to explain to people the novelty of informatics among scientific disciplines and its educational value. For this purpose, the formulation attributed to Cuny, Snyder and Wing[16] is appropriate: "CT is the thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent." This is almost the same definition given by Aho[1] "CT is the thought processes involved in formulating problems so their solutions can be represented as computational steps and algorithms" and Wing acknowledges the input received by him.[20] The big issue, as Armoni has clearly pointed out,[2] is that by taking any of these as the definition of a new discipline instead of as an explanation and trying to fully operationalize it causes more problems than benefits.

The issue of explaining in which sense "the way a computer scientist thinks" is different from "the way a mathematician thinks" is indeed an important one. Knuth had a brilliant example in his 1974 paper, which, unfortunately, is not at a level laypeople can understand. It regarded the problem of finding the "greatest common right divisor" of two $n \times n$ integer matrices A and B. The mathematician's answer is: "Let $\boldsymbol{R}$ be the ring of integer matrices; in this ring the sum of two principal left ideals is principal, so let D be such that $\boldsymbol{R}A + \boldsymbol{R}B = \boldsymbol{R}D$. Then D is the greatest common right divisor of A and B."[10] Clearly unsatisfactory for a computer scientist, for whom *a solution is provided by a process computing the answer and not by an equation defining the answer*. I have intentionally used the word "process" instead of the more usual "algorithm" to stress the fact that we have a "process" only when the algorithm has been implemented in a suitable "language" and an "automaton" executes the obtained code. In such a way three of the main pillars on which computer science is based—algorithm, language, and machine—are all involved in characterizing the difference between the viewpoints of the mathematician and the computer scientist.

I therefore think that, whenever either the Cuny, Snyder, and Wing's formulation or Aho's one is used for this explanatory purpose, the utmost stress must be put on the involvement of the *information processing agent* (that is, the "automaton," be it a machine or a person acting mechanically). *Without the agent and its capability to operate effectively, there is no informatics*, just mathematics, which indeed has been solving problems for millennia, discovering and applying along the way abstraction, decomposition, recursion, and so on.

Indeed, in looking backward toward how computer science was born, it is clear the cultural seeds are in the mathematicians' quest for automatizing theorem proving, in their efforts to unload the burden of solving problems onto machines. This shift in viewpoint, *from solving problems to having problems solved* is the intellectual birth of informatics and is the "difference which makes a difference,"[3] setting informatics in its own proper and unique place in the context of all sciences. The importance of the "automaton" to give full sense to CT was also made explicit[c] and emphasized.[1,6]

I also dare to provide, for the same demonstrative purpose, a more general explanation of what CT is, which is somehow along a direction already hinted at by Wing,[16] who clarified: "My interpretation of the words 'problem' and 'solution' is broad. I mean not just mathematically well-defined problems whose solutions are completely analyzable, for example, a proof, an algorithm, or a program, but also real-world problems whose solutions might be in the form of large, complex software systems." Nevertheless, Wing still used the word "problem," which conveys the meaning of something that needs to be solved.

Since solving a problem is just an instance of a situation where one wants to reach a specified goal, here is my formulation: *Computational thinking is the thought processes involved in modeling a* ***situation*** *and specifying the ways an information-processing agent can effectively operate within it to reach an* ***externally*** *specified (set of) goal(s).*" (See the accompanying figure.)

There are two main differences: one is speaking about a situation where the agent operates instead of a problem it has to solve, the other is clarifying the agent does not define by itself its overall (set of) goal(s) but gets it from the outside.[d] My formulation is also closer to more recent characterizations of computation as an unbounded process.[18]

c Aho wrote: "An important part of this process is finding appropriate models of computation with which to formulate the problem and derive its solutions." We could say, in a somewhat literary style, "the model is the agent is the model."

d If we allowed the agent to choose its own goals, we would leave computing and enter the realm of free-will entities.

We have thus a more general explanation of what CT is, covering also cases that are of high interest for schools and education: simulations in other disciplines, where one has to build and manipulate a visible representation of physical laws and/or natural/social phenomena (that is, to model a situation and explore its possible evolution) rather than to solve a problem. Simulation is a very powerful tool to improve understanding and computing is unique in its capability of making concrete the abstract models defined by a simulation.[2] In addition, we have a formulation that can be used to explain why mathematics or other sciences are not enough for these purposes.

In such a way informatics can more clearly explain its dual role[12] both as a *fundamental scientific subject*, with its own independent set of concepts, and as a *discipline of transversal value*, providing methods contributing to a better understanding of other disciplines.[7] This latter role of computing is also of particular importance for its introduction as a regular subject in schools, and can constitute a solid argument for considering it as a foundational discipline, on par with mathematics.[4,17] C

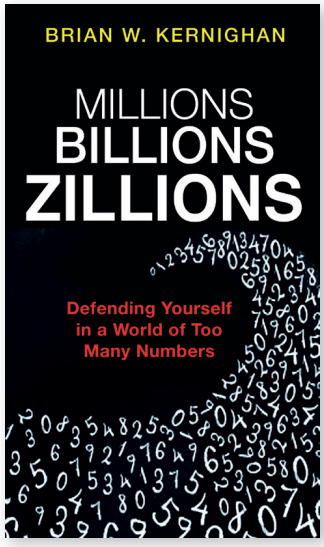
References

1. Aho, A.V. Computation and computational thinking. *Ubiquity*, vol.2011, issue January, Article no. 1, January 2011. ACM Press. DOI: https://doi.org/10.1145/1922681.1922682
2. Armoni, M. Computer science, computational thinking, programming, coding: The anomalies of transitivity in K–12 computer science education. *ACM Inroads 7*, 4 Dec. 2015), 24–27.
3. Bateson, G. Form, substance and difference. In *Steps to an Ecology of Mind*. University of Chicago Press, 1972.
4. Caspersen, M.E. et al. *Informatics for All: The Strategy*. ACM/Informatics Europe, NY, 2017; https://doi.org/10.1145/3185594
5. Denning, P.J. Computational thinking in science. *American Scientist 105*, (Jan.–Feb. 2017); 13–17.
6. Denning, P. Remaining trouble spots with computational thinking. *Commun. ACM 60*, 6 (June 2017), 33–39.
7. Denning, P.J. and Rosenbloom, P.S. Computing: The fourth great domain of science. *Commun. ACM 52*, 9 (Sept. 2009), 27–29.
8. Denning, P.J., Tedre, M., and Yongpradit, P. Misconceptions about computer science. *Commun. ACM 60*, 3 (Mar. 2017), 31–33.
9. Forsythe, G.E. What to do till the computer scientist comes. *The American Mathematical Monthly 75*, (May 1968), 454–462; https://bit.ly/2S19xXo
10. Knuth, D.E. Computer science and its relation to mathematics. *The American Mathematical Monthly 81*, 4 (Apr. 1974), 323–343; https://bit.ly/2ErRMMU
11. Lodi, M., Martini, S., and Nardelli, E. Abbiamo davvero bisogno del pensiero computazionale? *Mondo Digitale 72* (Nov. 2017), AICA, Milan; https://bit.ly/2CLJcr5
12. Nardelli, E. Informatica nella scuola: disciplina fondamentale e trasversale, ovvero "di cosa parliamo quando parliamo di pensiero computazionale." *Scienze e Ricerche Magazine* (Apr. 2017), 36-40; https://bit.ly/2GqszFk
13. Papert, S. *Mindstorms: Children, Computers, and Powerful Ideas*. Basic Books, 1980.
14. Polanyi, M. *The Tacit Dimension*. The University of Chicago Press, 1966.
15. Tedre, M. and Denning, P.J. The long quest for computational thinking. In *Proceedings of the 16th Koli Calling Conference on Computing Education Research*. (Nov. 2016), 120–129.
16. The LINK. Research Notebook: Computational Thinking—What and Why?. The Magazine of Carnegie Mellon University's School of Computer Science, March 2011; https://bit.ly/2UTeAed
17. Vahrenhold, J. et al. *Informatics Education in Europe: Are We All In The Same Boat?* ACM/Informatics Europe, NY, 2017; https://doi.org/10.1145/3106077
18. van Leeuwen, J. and Wiedermann, J. Computation as an unbounded process. *Theoretical Computer Science 429*, (2012), 202–212.
19. Wing, J. Computational thinking, *Commun. ACM 49*, 3 (Mar. 2006), 33–35.
20. Wing, J. Computational thinking benefit society. Social Issues in Computing blog, January 2014; https://bit.ly/2SOnisk
21. Wing, J. Computational thinking and thinking about computing, *Philosophical Transactions of The Royal Society A366*, 37 (2008): 3717–3725.

Enrico Nardelli (nardelli@mat.uniroma2.it) is a Full Professor in Informatics in the Department of Mathematics at the University of Rome "Tor Vergata," Italy. He is currently the president of Informatics Europe, the association representing the academic and research Informatics community in Europe.

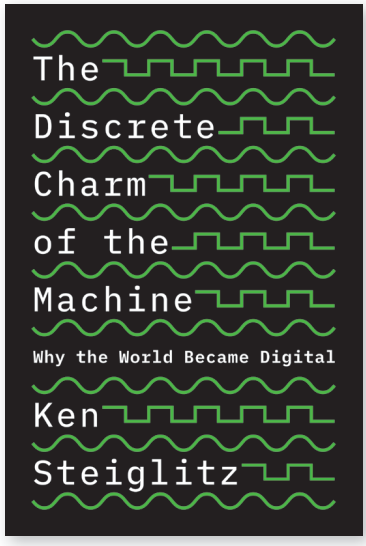
Discussions with Mehdi Jazayeri, Jan van Leeuwen, Michael Lodi, Simone Martini, and Guido Proietti have been useful to focus ideas and improve presentation; comments from referees have also been greatly helpful. Many of the ideas first presented in this Viewpoint have been further developed by the author in subsequent papers since this material was reviewed, revised, and accepted for publication in early 2017.

Copyright held by author.



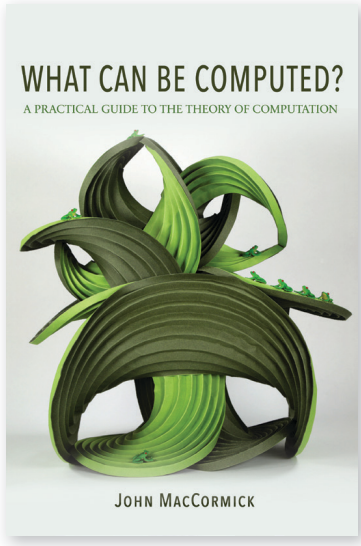
"The indispensable guide to numerical trickery, deception, and flimflam!"
—Harry Lewis,
coauthor of *Blown to Bits*

Cloth $22.95



"An inspirational must-read and delightful guide for anyone interested in traveling from the computational past through to the present."
—Andrew Adamatzky,
University of the West of England

Cloth $27.95



"*What Can Be Computed?* should succeed brilliantly in capturing the imagination of students."
—Matt Franklin,
University of California, Davis

Cloth $85.00

PRINCETON UNIVERSITY PRESS