Computational Thinking – What It Might Mean and What We Might Do About It

Chenglie Hu
Department of Computer Science, Carroll University
Waukesha, WI 53051, USA
1-262-524-7170
chu@carrollu.edu

ABSTRACT
Computational thinking has been promoted in recent years as a skill that is as fundamental as being able to read, write, and do arithmetic. However, what computational thinking really means remains speculative. While wonders, discussions and debates will likely continue, this article provides some analysis aimed to further the understanding of the notion. It argues that computational thinking is likely a hybrid thinking paradigm that must accommodate different thinking modes in terms of the way each would influence what we do in computation. Furthermore, the article makes an attempt to define computational thinking and connect the (potential) thinking elements to the known thinking paradigms. Finally, the author discusses some implications of the analysis.

Categories and Subject Descriptors
K.3.2 [Computer and Information Science Education]: Computer Science Education

General Terms
Theory

Keywords
Computational Thinking, Thinking Model, Computation, Computing Education

1. INTRODUCTION
Wing's influential article [18] suggested that computational thinking is a fundamental skill for us to gain understanding, live, and flourish in today's world. This promotion has been well received by the computing education community in the last few years, resulting in numerous workshops, conference panels and online discussions. Yet, the notion remains largely speculative today. Wing did not in fact define the term in her article. Indirectly, Wing described computational thinking to be involving solving problems, designing systems, and understanding human behavior by drawing on the concepts fundamental to computer science and by including a range of mental tools that reflect the breadth of the field. In another article [19], Wing refined the meaning of computational thinking: "It's a kind of analytical thinking, and it shares with mathematical thinking for problem solving, with engineering for modeling and design constrained by the real world, and with scientific thinking for understanding computability, intelligence, the minds and human behavior. The essence of computational thinking is abstraction that can be automated, which is what computing is about." These descriptions, and various others the author has seen, do not explicitly attribute the thinking ability to the (potential) thinking elements, structures, or traits. Indirect and often abstract characterizations of computational thinking have made the notion diversely interpreted.

Often, fruitful discussions can be more valuable than finding definitive answers. However, if being able to think computationally is indeed as fundamental as being able to read, write, and do basic math, then an entire K-12 education could be at stake if educators fail to reach a consensus on the notion and ways to teach it. While speculations, discussions, and debates will likely continue, this article intends to provide some analysis to articulate the potential nature of computational thinking. It also indicates some philosophical difficulties we may face when searching for more accurate descriptions of the notion. The analysis suggests that computational thinking is likely a hybrid thinking ability that people gain through a variety of means. The article then makes an attempt to conjuncture what computational thinking may mean and link the (potential) thinking elements to the better known thinking paradigms. Finally, the article discusses some implications of what has been analyzed. The article begins, however, by providing a brief review of a collective public perception of the notion that the author has found and studied.

2. DIFFERENT PERCEPTIONS ABOUT COMPUTATIONAL THINKING
Educators have given various descriptions of computational thinking over the last few years, although many dodged a direct definition. Their views were diverse and from sometimes very different assumptions, perspectives, and personal experiences.

The Chartered Institute for IT, formerly known as British Computer Society (at http://www.bcs.org/), held a recent BCS Thought Leadership debate aimed to discuss what computational thinking is from attendees' own experiences and how it impacts upon our everyday lives. Some participants believed that computational thinking helps determine what it is that can be computed, deal with systems that generate large amount of data,

ITiCSE'11, June 27–29, 2011, Darmstadt, Germany.

and better understand the constraints to a problem, computational limits and complexity. Others thought that computing and computational thinking collectively has changed science forever as computational modeling becomes a widely used tool within all disciplines. This, however, also works in reverse as the other sciences direct the way in which computational thinking is going. The participants felt that it's time for the IT community to convince other subjects that computer science is a subject in its own right, and not just a facilitator for others. Computational thinking is a common language to explore possibilities of computing, and a means to address a declining "sense of wonder" among digital natives. Yet, questions remained. Some participants concerned about teach-ability of the notion if computational thinking is about abstraction. Some suspected that abstraction may not be universally useful, and in certain situations such as language processing it can actually hinder understanding. Many wondered how computational thinking differs from the study of the conventional computer science subjects that has been fundamental to computing for many years, and how it differs from mathematical thinking that, too, deals with abstractions and representations. Some believed that many who are not computer scientists are actually doing computational thinking unknowingly if computational thinking is just a way of describing the dynamics and the processing of computing. Interestingly, some even wondered whether computational thinking was related to the tension between empirical and theoretical investigations, as such tension exists in other scientific disciplines.

To respond to the question "What is Computational Thinking?" posted at the Website of the Computer Science Teachers Association, a reader wrote: "I like to think of Computational Thinking as the ability to see, comprehend and devise systems and processes. For me this includes dealing with abstractions. The ability to see a solution to a problem as a 'process' is what makes all computation possible. To put it another way, if you have some ingenious solution to a problem but can't explain it as a process then you don't have a computational solution." (http://blog.acm.org/archives/csta/2009/11/what_is_computa.html) Computational thinking is also understood as interpreting and transforming data, for which a clear definition was given at http://gasstationwithoutpumps.wordpress.com/2010/08/12/algorithmic-vs-computational-thinking/: "Computational thinking is thinking about data by using computers to summarize, massage, or transform data into a more easily understood form. Contrast to computational thinking that focuses on the data and the interpretation of the data, the algorithms are just tools available to help with that focus."

Computational thinking is also perceived in some researches as how we do mathematics computationally. It is viewed, for instance, as an aid to modeling, representing, and solving mathematics problems, and effectively using mathematics in all other disciplines [13]. A word frequency scale, termed Computational Math Scale to measure the level of problem-solving gestalt exhibited in textbooks about computational mathematics, is described in [12]. To develop such a scale, researchers used books and articles exclusively in computational mathematics as artifacts of computational thinking. They then examined word frequencies in research articles and compared them to those that form the Computational Math Scale. They concluded that the words frequencies seem to suggest that Mathematical, Abstract, and Computational (MAC) thinking framework might integrate a wide range of topics relevant to computing.

Professionals at Google defined computational thinking to be thinking that involves a set of problem-solving skills and techniques that software engineers use to write programs that underlay the computer applications (http://www.google.com/edu/computational-thinking/what-is-ct.html). Meanwhile, notable computing educators also offered a rich set of opinions at a workshop organized by the National Research Council [14]. Collectively, they describe computational thinking as a form of procedural thinking; the study of the mechanisms of intelligence that can yield practical applications by magnifying human intelligence; the use of computation-related symbol systems to articulate explicit knowledge and manifest such knowledge in concrete computational forms; a way of formulating rigorous analysis and procedures for accomplishing a defined task efficiently; a meta-science to bridge between science and engineering; an open-ended and growing list of concepts that reflect the dynamic nature of technology and human learning; a careful reasoning about the methods of doing things, or thinking that complements mathematical or engineering thinking by combining the two.

The CPATH program of the National Science Foundation of the U.S. has supported endeavors aimed to promote computing education and foster computational thinking. Arguably, what people have done in CPATH-supported projects (the author has been involved in one such project) is not much different from what computing educators have done for years. Here is the dilemma. We seem confident that whatever we teach in computing promotes computational thinking. But why is this true? We struggle to answer this question. In fact, we don't seem to know the answers to some basic questions. Why is a separate promotion of computational thinking necessary given that it may share the thinking modes that are better known? To what extent would the notion really matter? Why would visual tools be the best way – or even a better way – to promote the learning of computational thinking and expect students to develop with the tools transferable skills of a higher order? And, what is computational thinking after all?

3. DIFFERENT THINKING MODES IN RELATION TO COMPUTING

A way of thinking conceivably consists of a set of thinking elements whether we realize or not. Foundation for Critical Thinking suggested one such collection of eight thinking traits that constitutes "critical thinking" at http://www.criticalthinking.org/courses/Elements_standards_model.cfm. To put them in a single sentence, whenever we think critically, we think with a purpose, raise questions, and embody a viewpoint by making assumptions and inferences and by using information and concepts, leading to implications. A person's critical thinking ability is applicable in any problem-solving context. Guided by the general principles of critical thinking, people in Computational X develop their own modes of thought commensurate with the kind of computation they do in the field X.

In computational physics, choosing a discrete model often requires a balanced consideration among physical constraints, numerical stability, accuracy, and computational cost. Physicists and numerical analysts contribute to finding a model by applying their distinctively different thinking modes. A physicist would

ensure that an intended numerical model is relatively faithful to physics, whereas a numerical analyst would have to study the computational ramifications of the model. Indeed, collaboration across disciplines is common in Computational X.

A way of thinking at a higher order generally requires systematic training in a relevant field. For instance, in theory, anyone who understands the classical Schwarz-Christoffel conformal mapping between two singly connected regions in the complex plane can apply a standard numerical quadrature to numerically compute the mapping function. However, not until the early 1980s had the numerical computation of the mapping function become successful [17]. Generally speaking, only people who are systematically trained in numerical analysis would be able to recognize and overcome obstacles that may hinder a successful numerical computation.

A way of thinking may not appear "computational", yet have significant computational ramifications. For instance, by asking insightful questions, applied mathematicians had successfully made connections between cellular automata and nonlinear dynamic systems of some kind to explore various dynamic properties of cellular automata and new rules of their computation [2]. They were able to bring the existing computational models of neural networks under a purely mathematical framework to study. Their findings had, in turn, influenced computation profoundly in terms of discovering much improved algorithms, leading to more efficient cellular neural networks.

In summary, people who use application software, develop the software, or study the models and algorithms are equipped with domain knowledge of different kinds that plays a pivotal role in a critical thinking process. Thus, computational thinking is likely diverse in nature, affording an ever growing scope. Certain ways of thinking in computing may require extensive training. Certain ways of thinking may not appear "computational" despite the (potential) computational implications. However, in the end, what makes computing meaningful, insightful, and fruitful is our collective thinking ability fueled with our distinct domain expertise and thinking modes.

4. MATHEMATICAL THINKING IN RELATION TO COMPUTING

Computing is more mathematical than many think it is. Some aspects of computing, similarly in mathematics, are about recognizing and manipulating patterns. Some others, such as software development, may need a certain degree of accuracy to measure the quality of what we do in various development processes even though we often struggle to find effective measurements. Programming constructs such as classes and objects are essentially mathematical entities that most who use them do not realize. And, programming, a significant form of computing, is arguably a mathematical activity [7].

However, computing appears much less mathematical than many think it should. Each computing area has its own methodologies that people may understand and be able to use with little knowledge of the underpinning mathematics. Indeed, doing computing can be rather accommodating. Writing a correct loop requires only thinking in algebraic terms. Yet, one can instead stack a sequence of statements to do the same thing if practical. People rely on running unit tests, not conducting correctness proofs, to show whether functional modules are algorithmically correct. Today, application programming interfaces, library frameworks, and enterprise-level integrated development environments are enabling people to develop software with little formal training. Poor thinking ability in abstract terms may be the reason why people are unable to produce clear, elegant designs and programs [8]. Thus, improving our mathematical thinking ability seems a logical way to improve the quality of what we do in computing.

Thinking mathematically appears also better understood. It may suggest the following thinking abilities [16]:

1. Exemplifying and specializing (in order to find examples of what is generally stated)
2. Completing, deleting, and correcting (in order to allow, ensure, or contradict conclusions to be made)
3. Comparing, sorting, and organizing (in order to better understand the assumptions and hypotheses)
4. Changing, varying, revising, and altering (questions, assumptions, hypotheses, constraints, or solution routes)
5. Generalizing and conjecturing
6. Explaining, justifying, verifying, convincing, and refuting (with consequences, extrapolations, reformulations, counterexamples, etc.).

A mathematical thinking process is also analytical to break a task down, make assumptions, identify similar tasks, appropriate knowledge and skills, look for patterns or connections, select a strategy while considering alternatives, and assist thinking with examples, data, or visual aids. Arguably, one does computing by taking full advantage of his or her mathematical thinking ability and the ability to follow a mathematical thinking process with, perhaps, a different orientation. In particular, thinking to model or design a system is a mental process to decompose the system into subsystems, conceptualize and simulate design choices, and apply convergent-divergent thinking cycles [5]. This is very similar to doing mathematics. Thinking mathematically directly translates into thinking recursively, abstractly, logically, and procedurally – the essential thinking abilities for anyone to do computing effectively. As said, mathematicians and computer scientists share several modes of thought, particularly in representation of reality, reduction to simpler problems, abstract reasoning, information structures, and algorithms [9]. Thus, the inseparability between mathematics and computing makes many wonder whether computational thinking is a form of mathematical thinking. However, one may also suspect that computing differentiates itself from mathematics with its unique orientation and intricacy, and hence may require more than just mathematical thinking. But how much is there in computational thinking that is different from mathematical thinking? If there is, would other thinking paradigms address the difference? We struggle to answer these questions.

5. IS COMPUTATIONAL THINKING A MIXTURE, PERHAPS?

In a way, various thinking paradigms may be related to one another "hierarchically". Thinking paradigms lower in the "hierarchy" are useful precisely because of their specificity. For instance, one tackles problems in a step-by-step fashion with refinement iterations by applying algorithmic thinking, which is likely a form of analytical thinking or mathematical thinking. In light of critical thinking, all thinking paradigms may share some common attributes disconnected to in-depth domain knowledge. Thus, a simple question such as "How many months are there in a year that each has 28 days?" may be used to test one's logical, mathematical, and perhaps, computational thinking abilities. But

evidently, it is the "high-order" computational thinking ability we are trying to understand.

It can, in fact, be rather philosophical to distinguish thinking paradigms when we engage in meaningful thinking. For instance, to improve performances of insertion and deletion operations of a sorted list, one wonders whether using a two-dimensional jagged array as storage might help (and it indeed will). Such thinking is clearly "computational". Can such thinking be labeled "analytical" or "mathematical" (and thus making "computational thinking" in such instances redundant)? It's difficult to argue it can't. Conversely, a philosophical undertaking, as described earlier, can impact computation profoundly, and yet may not appear "computational" at all. Arguably, one may apply essentially the same thinking process to produce possibly different mental products. For instance, mathematicians seek abstractions or representations to make mathematical structures richer, more predictable, or more complete. In contrast, computer scientists introduce abstractions or representations often for empirical reasons. However, they can be all using the same analytical thinking skill to create abstractions or representations and reason to seek them by exploring new ideas and approaches as they discard preconceived assumptions. In other words, one may acquire the very same thinking skill from a very different learning experience. Thus, it might be a philosophical challenge to stress the importance of computational thinking today while its products might have existed even before modern computers were born.

Nonetheless, "What might computational thinking be?" is at least philosophically interesting. To further the exploration, perhaps, we should have looked into the nature of computation in the first place. Classically, computation is an algorithmic process to produce output given input. Peter Denning describes computation as a process in which the transitions from one element of the sequence to the next are controlled by a representation [4]. Thus, he defines computational thinking to be an approach to problem solving that represents the problem as an information process and seeks an algorithmic solution. However, like many others, the definition might still be too broadly stated to be empirically helpful. What seems plausible however is the viewpoint that the essence of computation is to seek representations and models – the two intimately related yet subtly different concepts. Models, in a sense, are representations. However, a model – how entities in the model are represented – is a result of modeling, which is not simply how to represent things. Rather, modeling captures the dynamics of the entities based upon their representations. While a model can be abstract, a representation is most likely concrete. A model allows transforming data from one representation to another to make the data better understood or more "easily" manipulated. Models can be purely artificial, mathematically transformed, or algorithmically constructed by recognizing existing data patterns. With the above analysis, the author makes the following conjuncture of what computational thinking may mean in an operational sense.

Computational thinking is thinking to solve problems, automate systems, or transform data by constructing models and representations, concrete or abstract, to represent or to model the inner-working mechanism of what is being modeled or represented as an information process to be executed with appropriate computing agents. Such thinking is necessarily

- *logical, to capture what is essential to the models or representations;*
- *algorithmic, to step-wise define or refine operational processes;*
- *scientific, to gain understanding of models' capabilities, learn how to use them with maximum efficiency, and explore the effects of the computation in the original problem domain.*
- *mathematical, to be able to show the correctness of algorithms, specify precisely the functionality of a software system, measure the quality of what we do in a process of computation, and deal effectively with the complexity of the models and representations by exploring more effective and efficient alternatives;*
- *analytical, to model with purpose, assumptions and viewpoints, evaluate and adjust the models and representations by prototyping, and study their implications and consequences;*
- *engineering-oriented, to design the models and representations against known constraints and practical concerns, and to plan, execute, manage, and evaluate the process of computation in order to improve our capability and maturity level; and*
- *creative, to model the unthinkable.*

This definition is, in principle, consistent with the ones reviewed earlier. However, what makes this definition different is the linkage of the thinking elements to the better known thinking paradigms in terms of the relevance of each paradigm when applied to computation. Computation is diverse. Thus, it is hardly possible to describe computational thinking to encompass all possible thinking modes, their combinations and derivatives. The above definition should not preclude any thinking mode that can be more applicable in a specific area of computing with its focused characterization. For instance, software design needs "design thinking", thinking that enables a designer to tolerate ambiguity in an iteration of convergent-divergent thinking cycle, maintain sight of big picture, handle uncertainty, make sound decisions, and communicate in several design languages [5]. In the end, what matters is our ability to think critically, not the labeling of a thinking process.

Computing, as a discipline, has its well-established models of computation that we can still improve on. But perhaps, the perspectives drawn from other disciplines are really what makes computing full of wonders, challenges, and successes. Computation is unavoidable not only in the method of study, but in what is studied [3]. Likewise, computational thinking is present not only because of the nature of computation, but also because of the way how people think critically. We gain different kinds of critical thinking ability through a variety of means. These thinking skills, collectively, become a guiding force to enable us to do computation effectively. The more we do in computation, the more capable we are as computational thinkers. Thus, perhaps, it is not computational thinking we should promote, but computational doing at all levels of K-16 education in order for us to better understand the computational potential of the world in which we live. Therefore, whether or not we would be able to "accurately" define "computational thinking" might never be important.

6. IMPLICATIONS OF THE ANALYSIS

Colleges in the U.S. have faced years of declining enrollment in computing disciplines since the end of the dot-com era. The College Board of the U.S. has discontinued AP Computer Science AB Exam due to insufficient interests. Meanwhile, IT job markets remain strong in the U.S. and companies continue to seek IT talents overseas. The notion of computational thinking came in time to raise the level of urgency in promoting computing education across the entire spectrum of K-16 education.

Doing influences the way we think. A thinking paradigm means little to virtually anyone who hasn't had much experience in doing things with which the paradigm may help. We promote STEM education by having students solve STEM problems, not by advocating scientific, engineering, or mathematical thinking. There is inherently a C (Computing) in STEM. Learning STEM without learning computing is fundamentally inadequate. Learning computing while solving STEM problems, on the other hand, would inevitably foster one's computational thinking ability no matter how the notion is defined.

If the mainstream of computational thinking is thinking about process abstraction, then Jean Piaget's Stages of Cognitive Development [15] may suggest that this thinking skill cannot be effectively taught until adolescence age. Perhaps, what we need, instead, is a computational culture – a set of shared attitudes, values, goals, and practices that characterizes our education in which information processing and computation (in a variety of forms) are naturally integrated into what we teach. The authors of [11] proposed permeating the collective knowledge and lessons of computer science research into the discussion and development of all subjects that involve (information) processing. They also suggested some computational activities that can be naturally integrated into what we teach. In this way, students would be better prepared and more successful in learning programming as they progress computationally.

Fostering a computational culture is possible. When we ask students to search Web to find needed information, discuss how to do it effectively. When students are learning Excel software program, discuss ways to use it in solving perhaps optimization problems. When teaching students bisection method of finding roots of an equation, go a bit further to talk about binary search and other root-finding methods that can be potentially translated into more efficient search algorithms. When teaching polynomials, study the identity $a_nx^n + a_{n-1}x^{n-1} + \ldots + a_1x + a_0 = a_0 + x(a_1 + x(a_2 + \ldots + x(a_{n-1} + a_nx)\ldots))$ and its computational implications. Perhaps, students can learn how to design reasonably normalized databases as a modeling experience while still in high school. Students may possess commonsense computing abilities [10] that we should find ways to promote and build upon.

Yet, there are still serious obstacles. Studies have suggested that lack of or inadequate introduction of computer science at the high-school level, not the impact of the dot-com burst or IT overseas outsourcing, may have been a sustained major factor to prevent many capable high-school students from pursuing computing-related studies in colleges [1]. Meanwhile, teaching computer science is still an avocation, not exactly a hobby, but certainly not a primary job for many high-school computer science teachers [6]. As a result, few colleges have CS teacher education programs. To remove, or at least alleviate, the obstacles, educators have been promoting learning of CS in free environments (termed CS-unplugged at http://csunplugged.org/). They are developing a new AP course: "CS: Principles" (http://csprinciples.org) aimed to broaden participation in computing. But, it might require a pervasive plug-in in our curricula at all levels to eventually make computing a fundamental part of our education.

In closing, our ability to think critically and innovatively when we engage in computation will continue to improve as digital technology advances whether we promote computational thinking or not. In contrast, we are searching for means to improve our ability to make computing an integral part of K-16 education if promoting computational thinking can indeed help.

7. ACKNOWLEDGMENTS

The author would like to acknowledge the support of the NSF CPATH program No. 0939032.

8. REFERENCES

[1] Carter, L. Why students with an apparent aptitude for computer science don't choose to major in computer science. *SIGCSE 2006*, Houston, pp. 27-31.

[2] Chen, F. et al. Realization of Boolean Functions via CNN: Mathematical Theory, LSBF and Template Design, *IEEE, TRANSACTIONS ON CIRCUITS AND SYSTEMS—I: REGULAR PAPERS*, 53, 10 (October 2006), 2203-2213.

[3] Denning, P. Beyond Computational Thinking, *Commun. ACM*, Vol. 5, No. 6, June 2009, 28-30

[4] Denning, P. Ubiquity Symposium 'What Is Computation?' Opening Statement, Nov. 2010, http://ubiquity.acm.org/article.cfm?id=1880067

[5] Dym C. et al. Engineering Design Thinking, Teaching, and Learning, *Journal of Engineering Education*, January, 2005, 103-120

[6] Harrison, J. Endings and Beginnings, at http://blog.acm.org/archives/csta/2009/05/

[7] Hu, C. It's Mathematical, After All – the Nature of Learning Computer Programming, *Education and Information Technologies* (Springer Netherlands), 11, 1 (January 2006), 83-92.

[8] Kramer J. Is Abstraction the Key to Computing? *Commun. ACM,* Vol. 50 No. 4, April 2007, 37-42

[9] Knuth, D. Algorithmic Thinking and Mathematical Thinking, *The American Mathematical Monthly*, Vol. 92, No. 3 (Mar., 1985), 170-181

[10] Lewandowski, G. et al. Commonsense Understanding of Concurrency: Computing Students and Concert Tickets, *Commun. ACM*, 53, 7 (July 2010), 60-70.

[11] Lu, J. & Fletcher, G. Thinking about Computational Thinking, SIGCSE 2009, Chattanooga, PP260-264

[12] McMaster K. et al. Integrating Mathematical Thinking, Abstract Thinking, and Computational Thinking, *Proceedings of ASEE/IEEE Frontiers in Education Conference*, October 27 - 30, 2010, Washington, DC

[13] Moursund, D. *Computational Thinking and Math Maturity: Improving Math Education in K-8 Schools* (Second Edition), 2007, retrieved at http://uoregon.edu/~moursund/Books/ElMath/ElMath.html.

[14] National Research Council, Report of a Workshop on The Scope and Nature of Computational Thinking Committee for the Workshops on Computational Thinking, retrieved at http://www.nap.edu/catalog/12840.html

[15] Piaget, J. *Studies in Reflecting Abstraction,* Hove, UK: Psychology Press, 2001.

[16] Watson, A. & Mason, J. *Questions and Prompts for Mathematical Thinking*, Association of Teachers of Mathematics, Derby, 1998.

[17] Trefethen, L. Numerical Computation of the Schwarz-Christoffel Transformation, *SIAM J. Sci. Stat. Comput.* 1 (1980), 82-102.

[18] Wing, J. Computational Thinking, *Commun. ACM,* 49, 3 (March 2006), 33-35.

[19] Wing, J. Computational thinking and thinking about computing, *Phil. Trans. R. Soc.* A (2008) 366, 3717-3725