Computational Thinking, Redefined

Osman Yaşar
The College at Brockport, State University of New York
Brockport, NY 14420, United States
oyasar@brockport.edu

Abstract: The term computational thinking (CT) was popularized a decade ago as an "attitude and skillset" for everyone. However, since it is equated with thinking by experts in solving problems, more than a decade of discourse to capture its cognitive essence has resulted in a rather broad set of skills whose teaching to novices continues to pose challenges because of the reliance on the use of electronic computers and programming concepts that are often found too abstract and difficult by young students. The ongoing struggle in the field by teachers and educators on how to integrate CT practices and skills into K-12 education indicates that there still remain major trouble spots including definition, methods of measurement, cognitive aspects, and universal value of CT. This article redefines computational thinking by separating it from both the expert thinking and the use and programming of electronic computing devices. By merging concepts from epistemology, computing, cognitive and neurosciences, it introduces a framework which links CT to fundamental cognitive competencies that can be fostered at early ages using readily available tools in schools. The new framework also links computer science to natural sciences by putting computation at the heart of how everything forms and evolves in the universe.

Introduction

About forty years ago, Seymour Papert (1980) pioneered the idea of procedural thinking and programming for children through his LOGO programming project at MIT. However, it did not get much traction until Janette Wing (2006), then a division director at the National Science Foundation (NSF), championed computational thinking (CT) as a competency area using a general framework based on how computer scientists use computing to solve problems. It was a genuine and scholarly effort to link computing to fundamental functions of thinking at a time the field of computer science (CS) was suffering from low enrollments, thereby needing a radical reform to raise the interest and readiness of high school students. As such, it spread quickly as a concept and received national attention through federal funding programs, particularly NSF. Most of the literature on CT to this date still uses Wing's article as a springboard – according to Google Scholar it has been cited more than two thousand times! However, since a computer scientist employs a broad set of skills and strategies, this has made it hard to narrow CT down to a small set of competencies that everyone could agree on right away, let alone to teach it to everyone universally.

Workshops were encouraged by NSF to bring the community together in order to break down the meaning of CT. One of the early community-wide efforts that sought a consensus on the nature of CT and its cognitive and pedagogical implications was the two workshops by the National Research Council in 2010 and 2011. However, many questions remained unanswered. Given the concerns over the number of students seeking an education and a career in computer science, raising student interest and readiness in programming understandably had to be part of the overall enrollment initiative. So, while the workshops wanted the community to identify and share examples of how computing was used in solving problems by other sciences, the discourse could not break away from computer programming. Efforts continued diligently and persistently, including a revisit to the topic by Aho (2012) and Wing (2011) to clarify CT as the thought process carried out by an information-processing agent in order to formulate and solve a problem. Wing also stressed the need for CT research in learning sciences.

The CT initiative generated sizable momentum for teaching computational competencies in secondary schools. It even gained an international status. A pilot program was launched in 2012 to teach computing to all schoolchildren in England. In America, workshops were held to introduce computing concepts to college faculty and schoolteachers in order for them to test their applicability in the field. In 2011, the Computer Science Teachers Association (CSTA) explored operational definition of CT for K-12 teachers. A pilot AP course on Computer Science Principles was launched by the College Board (*http://www.csprinciples.org/*). And, finally, K-12 student learning standards in mathematics (*http://www.corestandards.org/math*), natural sciences (*http://www.nextgenscience.org*), and computer science (*https://k12cs.org/*) all started recommending the teaching of CT skills.

While periodic reviews, such as those by Grover & Pea (2013) and Denning (2017), on the status of CT education indicate wide agreement on what comprises CT, there is a struggle in the field by teachers and educators on how to integrate its practices and skills into K-12 education. Many researchers and educators who initially supported the idea of teaching CT skills to everyone have become wary of its promise. According to Denning, a prominent CS educator, some of the remaining trouble spots include definition, methods of measurement, cognitive aspects, and universal value of CT. This is no short order, and some of the letters to the editor, in response to his resent CACM paper (2017), suggest that unless somebody comes up with a more insightful definition, it may be time to retire "computational thinking" as a concept. To help ease such frustration in the CT education community, this article merges concepts from epistemology, computing, cognitive and neurosciences (Brown, Roediger, and McDaniel 2014; Hebb 1949; Montague 2006; Turing 1936; Yaşar 2016-2018) to introduce a new perspective that could address not only the remaining trouble spots of CT but also put computation at the heart of how everything behaves in the universe (Yaşar 2017b).

Current definition of CT and related skills

A recent article by Grover and Pea (2013) frames the current state of discourse on CT in K-12 education by reviewing recent academic publications. Denning (2017) offers an additional update by listing a growing number of various CT frameworks currently under consideration by the international community. These include operational definitions of CT by the Computer Science Teachers Association in 2011, The British Computer Society in 2015, and the International Society for Technology in Education in 2016. While there is a growing consensus as to what comprises CT and its curricular basis, no less than a dozen skills fall under these definitions as listed below.

Table 1: CT skills currently agreed upon by the computer science community.	
1. Logical reasoning	7. Collecting and analyzing data
2. Algorithmic thinking	8. Systematic processing of information (automation)
3. Iterative, recursive, and parallel thinking	9. Symbol systems and representations
4. Abstraction	10. Efficiency and performance constraints
5. Structured problem decomposition (modularizing)	11. Testing
6. Pattern generalization (including models & simulations)	12. Evaluation for efficiency and correctness

While these constitute a comprehensive set of basic skills in computer science, most of them are related to problem solving using electronic devices with an ultimate goal of preparing tomorrow's programmers. In fact, Grover and Pea (2013) argue "Programming is not only a fundamental skill of CS and a key tool for supporting the cognitive tasks involved in CT but a demonstration of computational competencies as well." (p. 40). They further claim that efforts such as CS Unplugged (*http: www.csunplugged.org*) that introduce computing concepts without the use of a computer may be keeping learners from the crucial computational experiences involved in CT's common practice. There is no scientific evidence to support this view, which maybe justifiably a mere reflection of the anxiety within the CS community about enrollments and its future as a natural field of science. In fact, to the contrary, the evidence suggests that teaching experts' habits of mind to novices is inherently problematic because of prerequisite content knowledge and practice skills needed to engage in the same thinking processes, not to mention the cost of providing them a similar environment to conduct inquiry and design (Kirschner, Sweller, and Clark 2006).

Much has happened to teach computing principles outside the context of programming but more than a decade of discourse and experimentation has yet to produce effective ways to separate CT from programming and the use of electronic devices. And, the lack of such separation continues to preclude us from capturing the cognitive essence of CT. Programming is probably the Achilles heel as far as preventing us from finding a simple and concise definition of CT in terms of basic fundamental skills that schoolchildren can learn and easily relate to without having to use computers initially. Programming electronic devices may be the central tool for computer scientists, but there are non-programming tools available to and used by other scientists who also use electronic computers. So, even if CT is tightly linked to problem solving with electronic computers, not every computer user solves problems the way a computer scientist does.

While the computer science community has recently expressed intent to generalize its original definition of CT to a thought process by an information-processing agent, be it a computer or any human (Aho 2012; Wing 2011), current curricular CT practices still continue to deal only with teaching of electronic CT skills. So, unless we reorganize the above CT skills into those that are cognitive in nature (i.e., biological CT skills) and those that are specific to use of an electronic computer device (i.e., electronic CT skills), we may still face a tall order to address remaining trouble spots described by Denning (2017). Thinking along those lines, here we suggest to organize them as shown in Figure 1 by assuming that information-processing by a device, be it electronic or biological, will have device-independent aspects. Then, to help realize a universal CT reform that everyone can benefit from, we educators can undertake two sequential steps to teach CT skills. We can first foster cognitive skills that are common to both biological and electronic computing devices, followed by others that are related to use of electronic computing devices alone. This would still leave us a task of narrowing down and simplifying electronic and biological CT skills so they can be taught to novices, but it would at least give us a framework to agree on a universal definition for CT – *i.e., thinking generated and facilitated by computation, regardless of the device that does the computation* – and seek a device-independent computational methodology/tool to facilitate both cognitive (biological) and electronic CT skills.

Device-independent storage, retrieval, and processing of information

The attempt to link electronic and biological (i.e., cognitive) computing goes back to Alan Turing (1936), the founder of computer science, who suggested that information processing by a device would depend on the nature of information that needs processing as well as the hardware that does the processing. His idea that our thoughts consist of quantifiable constructs of information suggested that any computing device, be it electronic or biological, could add, subtract, and re-arrange them as our brains do. While each device would have a different hardware to do this, a similarity (i.e., a device-independent pattern) can be expected in their processing because of an invariant computable (addition and subtraction) nature of information.

Electronic machines have since taken many complex and voluminous computations off our brains, further supporting the cognitive science view of brain as a biological computational device (Montague 2006). While such a view has yet to fully capture complex mental representations and emotions of our brains (Goleman 2006), recent imaging techniques in neurosciences continue to help us understand how at least the brain stores, retrieves and processes information from electrical activities of a distributed structure of neurons.

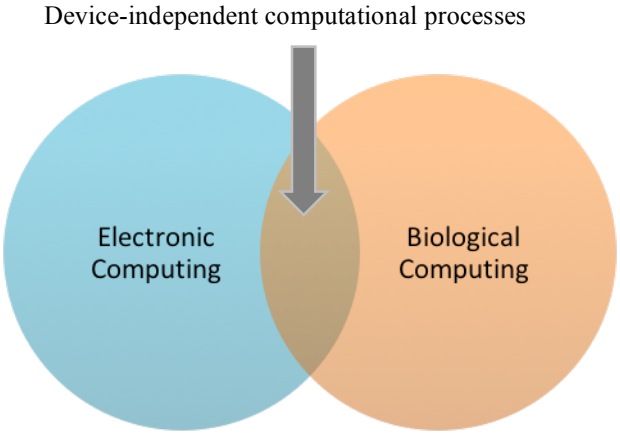

Figure 1: Information processing by electronic and biological computing devices include both device-independent and device-dependent processes.

Basically, information gets stored into the memory in the form of a specific pattern of neurons placed on a pathway and fired together (Hebb 1949). A memory or a newly learned concept can be a combination of previously formed memories, each of which might also involve a vast network of concepts and details mapped onto the brain's neural network in a hierarchical way as shown in Figure 2. The key to storing a concept more permanently into the memory is, then, to link it to previously stored basic and retrievable concepts. When new information arrives, it lights up all related cues, neurons and pathways in a *distributive* process that is similar to the top-down action in Fig. 2, where a new concept is broken up into related pieces. With the same token, retrieving a memory is a reassembly of its original pattern of neurons and pathways in an *associative* process that is similar to the bottom-up action in Fig. 2.

Neuroscientists now see little or no distinction between the act of thinking and the acts of information storage and retrieval (Brown, Roediger, and McDaniel 2014). If so, Turing's idea that the act of thinking involves information processing for (distributive) storage and (associate) retrieval of thoughts and concepts has more support today than ever before. The brilliance of it, of course, was the consideration of information as consisting of quantifiable constructs, just like the granular matter. It appears, then, that our brain's inclination to store, retrieve, and process

information in an *associative/distributive* fashion is just a manifestation of a duality in the nature of quantifiable information. Such duality may simply be caused by the dual (quantifiable) behavior of the matter that the sensory information is a reflection of. Whatever the underlying cause, our brain's cognitive inclination is probably an evolutionary response, shaped up over many years, to optimize its handling of sensory information whose quantifiable nature only resonates with distributive and associative operations. We can see a similar evolution in electronic computing. Turing's idea of an electronic device to imitate the biological brain has indeed evolved quite dramatically since its first design; particularly in regard to decentralization of information processing and storage. For example, today's electronic computing devices process and store information in a distributed way, somehow similar to the distributed brain circuitry. Accordingly, programmers of parallel computers know that management and utilization of a distributed hardware necessitates *scatter* and *gather* type communication functionalities in software and that is also similar to how the distributed neural structure stores and retrieves data.

While the brain does not work exactly like today's electronic computers, it is important to note two natural sources of their similarities. One is obviously the deliberate design, use, and control of electronic computing devices by biological computing agents. The other is the appearance of a computational (associative/distributive) process (Fig. 2) with which information constructs conform (Yaşar 2017a). So, be it electronic or biological, computing devices are likely to use similar ways to track and tally the invariant behavior of information constructs. In fact, as we will see in the next section this device-level pattern of computation carries itself up to higher levels of computational and cognitive processing. One could argue, then, one of the best ways an electronic computing device can benefit its human user cognitively is to help him/her with associative/distributive processing of information. A simple form of such help has been basic computations (addition and subtraction of numbers) that we already do with our calculators. There isn't much cognitive benefit from that, obviously, and some of us could do it without such tools. Yet, one wonders if there are other higher-level associative/distributive computations we can delegate that are known to facilitate thinking. The answer is "Yes," and luckily just like calculators, they are available to novices these days. To identify, we need to review some common concepts in cognitive sciences and epistemology as summarized below.

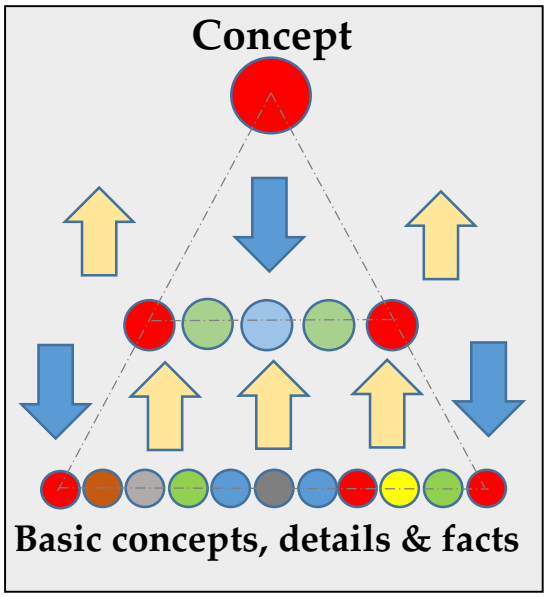

Figure 2: A mechanism by which information (and the granular matter) form and evolve (Yaşar 2017a).

Cognitive view of thinking and learning

While the distributed structure of neurons and their connections (i.e., hardware) influence cognitive processing (i.e., software), the relationship between software (mind) and hardware (brain) is not a one-to-one relationship. Accordingly, our mind consists of a hierarchy of many patterns of information processing and, just as it is the case in electronic computing, these levels may range from basic computations to more complex functions (sequence or structure of instructions) and models (mental representations) of perceived reality and imaginary scenarios (Montague 2006). As such, when we encounter a new concept and information, we attempt to interpret it in terms of previously registered models ─ *objects, faces, scenarios, etc.* This is known as deductive reasoning, or an act of analysis, and it is consistent with how neurons fire distributively (similar to top-down action in Fig. 2). As the time goes, the relationships among registered information constructs lead to interplay of various combinations and scenarios that eventually end up clustering related details into conclusions, generalizations, and more inclusive constructs of information in an associative fashion (similar to bottom-up action in Fig. 2). This is called inductive reasoning, or an act of synthesis. Eventually, the details our brain registers and stores and the hierarchical connections it establishes between them, along with these generalizations and conclusions, build over time like a pyramid-like structure that we have come to call *mind* (Bransford, Brown, and Cocking 2000).

Neuropsychologists and evolutionary biologists point out to a structural (hardware) factor that makes us appear to be caught between two brains (Evans and Frankish 2009) with opposite tendencies to control our decision-

making process. Our autopilot limbic system (animal-like brain) appears to be interfering with the outer parts of our brains (neocortex) to by-pass, simplify, or even reduce its elaborate cognitive functions. It is as though the limbic system wants to simplify things for quicker and automated decision-making while the cortex wants to dig things deeper for more informed and yet slower decision-making. Cognitive scientists point out rather to non-structural (software) factors to explain this competition. For example, Montague (2006) suggests that concern for efficiency, as part of our survival, is a major factor. He argues that we assign value, cost, and goals to our thoughts, decisions, and actions, and we choose among them those that save us energy and help our survival. To assigns these attributes, the mind carries out computations, builds models, and conducts hypothetical simulations of different scenarios before it decides what to do. Whether the causes are structural or non-structural, there is enough evidence to suggest an overriding cognitive inclination to either simplify things or to dig them deeper. An underlying information-processing mechanism to derive such cognitive functioning appears to be the very associative/distributive pattern in Fig. 2 that we discussed in the previous section as being an outcome of the quantifiable nature of information.

Modeling & Simulation: A high-level device-independent process

To optimize our response to (and survival against) a changing environment, our need for simplicity and generalization via inductive reasoning (which employs associative processing of information) as well as for complexity and details via deductive reasoning (which employs distributive processing of information) has pushed us into an iterative and cyclical use of them which has over time driven us to think harder and become smarter. We use inductive reasoning to filter out details and place our focus on more general patterns, thereby assigning priority and importance to the newly acquired information. We also use deductive reasoning to help us make decisions and draw conclusions from general concepts. There are many benefits of each, but since they require effort, the efficiency, intactness, and effort-fullness with which we all use them iteratively and cyclically would depend on the individual. A scientist is a good example of someone who employs them in a more frequent, consistent, and methodological way than most people (Kant 1787). Scientists have enjoyed this methodology for several centuries, some more than others, but today's easy-to-use technology is now slowly bringing the same culture and benefits to a wider audience.

Modeling has been an important tool for scientific research for hundreds of years. In principle, it works exactly as illustrated in Fig. 2. Scientists ideally start with a model (concept/theory) of reality based on current research, facts, and information. They test the model's predictions against experiment. If results do not match, they, then break down the model *deductively* into its parts (sub models) to identify what needs to be tweaked. They retest the revised model through what-if scenarios by changing relevant parameters and characteristics of the sub models. By putting together *inductively* new findings and relationships among sub models, the initial model gets revised. This cycle of modeling, testing, what-if scenarios, synthesis, decision-making, and re-modeling is often known as *conceptual change* and the process is repeated while resources permit until there is confidence in or satisfaction with the revised model's (or concept's) validity. Electronic computers have recently accelerated this cycle because not only do they speed up the model building and testing via simulations but they also help conduct studies that are impossible experimentally due to size, access and cost. No wonder that the scientific progress of the last 50 years far exceeds that of the previous several centuries.

Modeling and simulation (M&S) appears to be a high-level device-independent process of information that links computing and cognition. Its inductive/deductive reasoning mechanism has been used by biological computing agents for ages and by the electronic computing devices whose design/use/control by some of those agents have evolved, in the past 50 years, to a point of making Turing very proud. Some may argue that part (or all) of the credit should go to the nature itself because it is the heterogeneity of matter (and information) that has made such a dynamic evolution possible. There will certainly be some benefit to the CS community to engage in a discourse with other sciences to link computation to nature's inherent behavior. M&S's associative and distributive processes are indeed so universal that they describe computable actions of all quantifiable things. For example, formation of physical objects or particles from smaller ones resembles the act of modeling because both seem to involve packing *parts* together associatively to form a *whole*. And, such act of modeling is often driven by external forces or by a collective "trial and error" process controllable by conditions and rules of engagement ─ much like a simulation.

Philosophers and psychologists have been studying the *parts-and-whole* dynamics since Plato (Findlay and Thagard 2012) to explain the nature and human behavior. Recently, with help from technology, cosmologists and cognitive scientists have also been searching for a universal process that may be guiding the growth of all networked systems, ranging from the tiny brain cells to atoms, to the Internet, and even the galaxies. The view that such a process

may be described computationally, as in Fig. 2, is now gaining traction because formation and evolution of an abstract idea or a computational model of information appears to be no different than that of a system of physical particles (Yaşar 2017a-b). If so, then not only can we learn from an ongoing millennial argument of such a universal topic, but we can also put computing at the center of a discourse well beyond CT to understand the nature itself.

The essence of computational thinking

As suggested earlier, we need to distinguish between electronic and biological CT skills in order to more effectively teach each at relevant grade levels. Figure 3 illustrates this relationship (Yaşar 2017a). Since associative/distributive way of processing, storing, and retrieval of information appears to be the essence of thinking generated and facilitated by a computational mind, we put this duality at the core of information processing involved in both electronic and biological computing skills. Such processing of information at the most basic level involves fundamental computations (+ and -) of simple information constructs. At higher-levels, the constructs become larger and more complex but the operations and algorithms to manipulate these constructs still build upon the most basic computations at the device level.

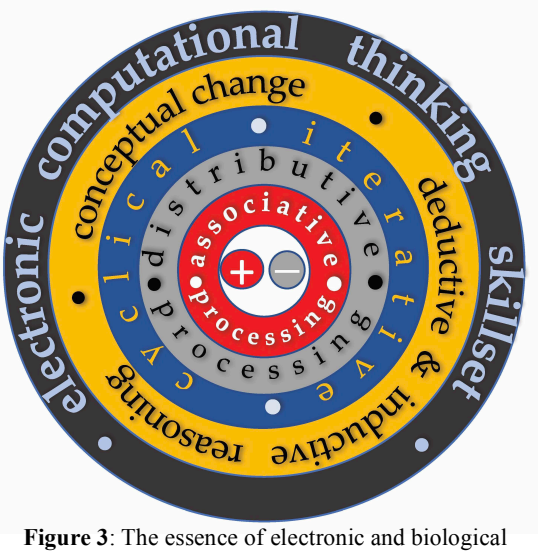

Figure 3: The essence of electronic and biological CT skill sets (Yaşar 2017a).

Since M&S is considered to be common to both electronic and biological (cognitive) computing (Montague 2006), knowingly or unknowingly everyone employs its associative/distributive processing (Fig. 2) in order to make decisions. While cognitive scientists agree that this type of information processing facilitates inductive reasoning (i.e., synthesis) and deductive reasoning (i.e., analysis), its full potential actually lies in an iterative and cyclical use of these opposite processes for learning progression and conceptual change. If one's use of these mental processes can be facilitated by the use of electronic M&S tools during teaching and learning, then such potential may be tapped into more fully. Empirical data in the past decade supports this assertion as discussed in the next section.

All in all, the hierarchy shown in Fig. 3 illustrates the core essence of computational thinking that we all employ for learning, conceptual change, scientific inquiry, problem solving, and even everyday decisions (Bransford, Brown, and Cocking 2000; Montague 2006). While everyone, not just computer scientists, does this kind of computational thinking for survival on a needs base, its utilization will depend on the underlying device structure and the quality and quantity of the environmental input it receives. The desired situation would be that one employs the processes of analysis and synthesis more frequently, effectively and methodologically to make better and more informed decisions. Those who wanted to further facilitate such cognitive functions certainly have used other computing devices in the past, and the use of computer modeling and simulation by researchers reflect such a use. Yet, these researchers already had a habit of using those cognitive functions. Accordingly, prior to teaching students electronic CT skills (the outer circle in Fig. 3), we first need to teach them a cognitive habit of conceptual change that is facilitated by an iterative and cyclical use of inductive and deductive reasoning. This is not a short order either but it at least puts the horses ahead of the carriage.

Some elements of the electronic CT skillset in Table 1 naturally overlap with the fundamental cognitive processes shown in Fig. 3. Of all the skills in Table 1, only those on the left column appear to have a cognitive correspondence in biological CT, as we defined here. Those on the right column can be considered as device-dependent skills resulting from the use of an electronic device for problem solving by a biological computing agent. Of those on the left, the two most essential cognitive and electronic CT skills are *abstraction* and *decomposition* because they directly involve the kind of associative and distributive processing of information that we discussed above. Abstraction is an *inductive* process which helps our cognition, especially at its developmental stages, by simplifying, categorizing, and registering key information and knowledge for quicker retrieval and processing (Bransford, Brown, and Cocking 2005). Decomposition is a deductive process which helps us deal with a complicated situation, by dividing the complexity into smaller and simpler pieces and then attack each one separately until a

cumulative solution is found. If abstraction is considered as packaging (modeling) of things, then decomposing (analyzing) a concept, a package, or a model deductively is as important as constructing it inductively. They are equally essential, and we use them in our daily lives but not everyone is equally aware of their importance, nor are we all practicing and utilizing them fully and equally. But, since abstraction and decomposition skills are heavily used in programming and problem solving having students improve them has been a concern of educators (Armoni 2013). A good programmer is expected to be able to oscillate between various levels of abstraction (see Fig. 4) and employ decomposition skills to divide the problem and programming into functional and modular units. According to some CS educators, undergraduate CS students barely move beyond language-specific (level 2) or algorithm-specific (level 3) biases (Armoni 2013). While reaching level 4 is important to transform appropriate algorithmic and programming skills into different application contexts, the teaching of programming itself does not seem to accomplish this. Students would gain more from their education if, prior to taking programming courses, they appreciate the importance of inductive thinking for abstraction and of deductive thinking for decomposition skills in their assignments and large and complex software projects. It appears that such appreciation can be gained at the secondary school level as some of the recent curricular CT practices have shown (Yaşar et al., 2014).

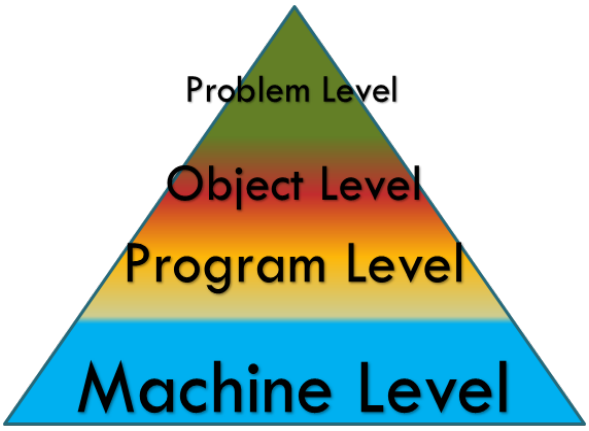

Figure 4: Multi-level abstraction in programming, From low to high: (1) machine-level, (2) program-level, (3) object-level, and (4) problem-level).

CT Education Research

In addition to the nature, cognitive essence, and universality of CT, it is important to address some of the implementation and curricular issues. Luckily, there is a tight link between the iterative and cyclical process of inductive/deductive reasoning and the thought process involved in *conceptual change* – an important element of what has been known as the scientific method or recently referred to as scientific thinking (ST) (Vosniadou 2013). So, fostering such fundamental ST skills ought to come before an electronic CT education. While ST education research goes back decades and even centuries earlier than CT education, teaching expert-level ST skills to novices have also run into similar roadblocks (Dunbar & Klahr 2012; Yaşar, Maliekal, Veronesi, and Little 2017). We should be able to learn much from the ST education community, particularly in regard to conceptual change aspect of ST because it is an essential element of thinking and learning progression, regardless of the area of knowledge that uses it. In fact, biological computational thinking, as we discussed here, is the essence of expert thinking not only in computing but also in other areas such as scientific thinking and engineering thinking (Yaşar *et al.* 2017).

The science education community has considerably researched effectiveness of M&S on learning and this also makes us lucky if were to use such tools in CT education. A literature review by Smetana & Bell (2012) identifies M&S as an exemplar of inquiry-guided learning. Its effectiveness is also well grounded in contemporary learning theories that recognize the role of abstract thinking and reflection in constructing knowledge and developing ideas and skills (Bransford, Brown, and Cocking 2000). Since these theories suggest that students learn better when they are engaged in activities closely resembling the way scientists think and work, it should not come as a surprise that M&S helps students learn as well. Additional details about pedagogical aspects of M&S, as supported by a quasi-experimental study, can be found in (Yaşar 2016; Yaşar *et al.* 2016; Yaşar *et al.* 2015; Yaşar and Maliekal 2014).

While educational researchers have done a good job of measuring the impact of M&S on learning, there is a need to measure its impact on conceptual change, abstraction, decomposition, and metacognitive skills, particularly in relation to CT and programming education. Besides M&S tools, researchers should explore other modular and scalable design toys as well as reading and writing practices to offer similar CT practices. There are many instruments with good psychometric properties to measure the impact of technological pedagogical content development tools on teaching and learning (Koehler, Shin, and Mishra 2012). M&S's interdisciplinary and changing technological nature require customization of its use in instruction and the assessment of its effectiveness in teaching of the content under consideration. Researchers may need to use not only quantitative methods to measure variables involved but also qualitative methods to initially identify those variables and to later understand and triangulate them for validity. The

quantitative sources of data often include surveys to gather pre/post activity data, unit test scores, course passing rates, report cards, graduation rates, and achievement scores in standardized tests) while qualitative sources of data may be interviews, classroom observations, and computational artifacts (Fincher and Petre 2005).

Conclusion

The discourse on what constitutes CT and how it can be taught and evaluated still goes on. The anxiety and enthusiasm that initially fueled the promise and spread of CT reform, the ongoing pressure by government agencies and professional organizations to implement it, and the remaining confusion by teachers and educators to bring a closure to or retire the concept of CT altogether warrant a new perspective. A decade ago, prominent educators inductively arrived at a concept/model of what comprises CT which obviously did not fully stand the test of time. As a result, many patches have been offered over the years that are not fully satisfactory to some who are invested in it. A product of decades of research and education in computing and cognitive sciences, the CT perspective and framework articulated in this paper has deductively broken down the current concept of CT and re-modeled it iteratively and cyclically to bring much-needed coherence, simplicity, and relevance to realize its universal promise. The cognitive CT framework presented here as a whole is based on the sum of several parts, including Kant's epistemological method (1787), Turing's computational theory of mind (1936), and Hebb's neuropsychological view of memory storage and retrieval (1949). As Koffka (1935) noted, when put together parts generate a *holistic* view that is not only more than the sum of the parts but also *other* than any view that each part may convey alone.

While the computational theory of mind may not fully capture complex mental representations and emotions, it helps us to understand some fundamental aspects of both ordinary and expert thinking. A computational mind equips us to process incoming sensory information in two distinct (and opposite) ways, thereby setting us up for an evaluative decision-making which requires assigning value, priority, and cost to all potential paths and actions we can take. To meet this requirement, the mind carries out computations, builds models, and conducts evaluative and hypothetical simulations of different scenarios. Everyone benefits cognitively from associative/distributive processing of modeling and simulation by the virtue of having a computational mind that employs inductive/deductive reasoning. Some use it in a more systematic way in their lives and some use it in a more automated way in their professions with the help of electronic or other computing devices.

The computer science community has traditionally kept a distance to computational M&S because of its focus on programming. This caused it to miss on being a key player in development of computational science programs in the 80s and 90s by leaders of physical and life sciences (Denning 2009). At the same time, despite its importance computational M&S education did not grow beyond a few PhD and master's programs within science and engineering fields, either, because of needed expertise in computer programming. However, both the CT initiative by the computer science community and the availability of non-programming M&S tools have recently created an opportunity for computational skills education to move all the way down to the middle and even elementary schools. As such, many K-12 STEM courses have already been using M&S tools and the national learning outcomes are now recommending the teaching of computational thinking skills. To help overcome the trouble spots, though, computing, cognitive, physical, and life sciences need to join hands together. The CT framework described in this article and the relevant publications below offer additional information for researchers as well as teacher and technology educators who may want to test or implement curricular resources developed by the author and his colleagues. Currently, these resources are being utilized at a rate of 80-100 downloads per day (see http://digitalcommons.brockport.edu/cmst_institute/).

As a final note, computation, and particularly M&S, appears to have a universal value far beyond its relation to cognition. In fact, M&S's iterative and cyclical mechanism of associative and distributive processing appears to be the essence of natural dynamism of all discrete forms (Yaşar 2017b). By putting computation at the heart of natural sciences, M&S provides an opportunity for us to claim that computer science deals with natural phenomena, not artificial (Denning 2009). So, a call for action here for the CS community is to put more emphasis on M&S as a crucial part of student practice and education. Since it facilitates an iterative and cyclical process of deductive and inductive reasoning, it could definitely be used to teach novices better abstraction and decomposition skills. Some studies have already shown that it not only teaches students such CT skills but it also motivates them to learn computer programming (Yaşar & Maliekal 2014).

References

1. Aho, A. (2012). Computation and Computational Thinking. *The Computer Journal*, 55(7); 832-835.
2. Armoni, M. (2013) On Teaching Abstraction to Computer Science Novices. *J. Comp in Math & Science Teaching*, 32(3); 265-284.
3. Bransford, J., Brown, A. and Cocking, R. (2000). How People Learn. National Academy Press, Wash., D.C.
4. Brown, P. C., Roediger, H. L. & McDaniel, M. A. (2014) *Make it Stick*. Belknap Press.
5. Denning, P. (2017) Remaining Trouble Spots with Computational Thinking. *Comm. of the ACM*, 60 (6); 33-39.
6. Denning, P. (2009). Beyond computational thinking. *Comm. of the ACM*, 52 (6); 28-30.
7. Dunbar, K. & Klahr, D. (2012). Scientific Thinking and Reasoning. In K. Holyoak and R. Morrison (Eds.), *The Oxford Handbook of Thinking and Reasoning* (pp. 701-718). London: Oxford University Press.
8. Evans, J. and Frankish, K. (2009). *In Two Minds: Dual Processes and Beyond*. Oxford University Press: Oxford.
9. Fincher, S. and Petre, M. (2005). *Computer Science Education Research*. Taylor and Francis e-Library.
10. Findlay, S. D. and Thagard, P. (2012). "How parts make up wholes." *Frontiers in Physiology*, 3, 455.
11. Goleman, D. (2006). *Emotional Intelligence.* New York: Bantam Dell
12. Grover, S. & Pea, R. (2013). Computational Thinking: A Review of the State of the Field. *Educational Researcher,* 42(1); 38-43.
13. Hebb, D. (1949). *The Organization of Behavior*. New York: Wiley & Sons.
14. Kant, I. (1787). *The Critique of Pure Reason*. (J. M. D. Meiklejohn, Trans.). eBook@Adelaide, The University of Adelaide Library, Australia.
15. Kirschner, P. A., Sweller, J. & Clark, R. E. (2006). Why Minimal Guidance During Instruction Does Not Work. *Educational Psychologist.* 41 (2), 75-86.
16. Koehler, M., Shin, T., & Mishra, P. (2012). How do we measure TPACK? In R. N. Ronau, C. R. Rakes, & M. L. Niess (Eds.), *Educational technology, teacher knowledge, and classroom impact* (pp. 16-31). Hershey, PA: IGI Global.
17. Koffka, K. (1935). Principles of Gestalt Psychology. New York: Harcourt, Brace, and World.
18. Montague, R. (2006). How We Make Decisions. Plume Books: New York.
19. Papert, S. (1980*). Mindstroms: Children, computers, and powerful ideas*. New York: Basic Books
20. Smetana, L. K. and Bell, R. L. (2012). Computer Simulations to Support Science Instruction and Learning: A critical review of the literature. *Int. J. Science Education*, 34 (9); 1337-1370.
21. Turing, A.M. (1936). On Computable Numbers, with an Application to the Entscheidungs problem. *Proc. of the London Mathematical Society.* 2 (1937) 42: 230–265.
22. Vosniadou, S. (2013). *International Handbook of Research on Conceptual Change.* 2nd Edition. New York and London: Routledge.
23. Wing, J. M. (2011). Computational thinking – What and why? *The Link Magazine*. March 06, 2011.
24. Wing, J. M. (2006). Computational Thinking. *Comm. of the ACM*, 49(3); 33-35.
25. Yaşar, O. (2018). A new perspective on computational thinking. *Comm. of the ACM,* In Press.
26. Yaşar, O. (2017a). The essence of computational thinking. *Computing in Science & Eng.*, 19 (4); 74-82.
27. Yaşar, O. (2017b). Modeling & Simulation: How Everything Seems to Form and Grow. *Computing in Science & Eng.*, 19 (1); 74-78.
28. Yaşar, O., Maliekal, J., Veronesi, P. and Little, L. (2017). "The essence of scientific and engineering thinking and tools to promote it." *Proc. Am. Soc. Eng. Education Annual Conf.*, Columbus, OH.
29. Yaşar, O. (2016). Cognitive Aspects of Computational Modeling & Simulation. *J. Computational Science Education,* 7(1), 2-14.
30. Yaşar, O., Veronesi, P., Maliekal, J., Little, L., Vattana, S., and Yeter, I. (2016). Computational Pedagogy: Fostering a New Method of Teaching. *Comp in Education,* 7(3), 51-72. Received a Best Paper Award at America Society of Engineering Education (ASEE) Annual Conference in 2016, New Orleans, LA.
31. Yaşar, O., Maliekal, J., Veronesi, P., Little, L. and Vattana, S. (2015) Computational Pedagogical Content Knowledge. In L. Liu and D. C. Gibson (Eds), *Research Highlights in Technology and Teacher Education*. pp. 79-87. ISBN: 978-1-939797-19-3. Received Anne Thompson TPACK Paper Award at the Society of Information Tech. & Teacher Education (SITE) Annual Conf. in 2015, Las Vegas, NV.
32. Yaşar, O. and J. Maliekal (2014). Computational Pedagogy. *Computing in Science & Eng.*, 16 (3); 78-88.
33. Yaşar, O. & Maliekal, J., Veronesi, P. and Little, L. (2014). An Interdisciplinary Approach to Professional Development of Math, Science & Technology Teachers. *Comp. in Math & Sci. Teaching*, 33(3), 349-74.