Journal for STEM Education Research (2020) 3:1–18
https://doi.org/10.1007/s41979-020-00030-2

EDITORIAL

Computational Thinking Is More about Thinking than Computing

Yeping Li[1] · Alan H. Schoenfeld[2] · Andrea A. diSessa[2] · Arthur C. Graesser[3] · Lisa C. Benson[4] · Lyn D. English[5] · Richard A. Duschl[6]

Published online: 18 May 2020

Abstract

Computational thinking is widely recognized as important, not only to those interested in computer science and mathematics but also to every student in the twenty-first century. However, the concept of computational thinking is arguably complex; the term itself can easily lead to direct connection with "computing" or "computer" in a restricted sense. In this editorial, we build on existing research about computational thinking to discuss it as a multi-faceted theoretical nature. We further present computational thinking, as a model of thinking, that is important not only in computer science and mathematics, but also in other disciplines of STEM and integrated STEM education broadly.

Keywords Cognition · Computational literacy · Computational thinking · Computing · Models of thinking · STEM integration

Introduction

In our second joint editorial (Li et al. 2019a), we focused on design and design thinking in science, technology, engineering and mathematics (STEM) education, and discussed design thinking as an example of models of thinking that are important to each and every student. Contrary to a common perception that design and design thinking belong to certain subjects but not others, we highlighted the need and significance of changing the *subject fixation* perception to elevate the conception of design thinking as transdisciplinary and not belonging to the field of engineering only. Based on our proposed notion of "everyone designs and can design" (Li et al. 2019a), we further discussed

Electronic supplementary material The online version of this article (https://doi.org/10.1007/s41979-020-00030-2) contains supplementary material, which is available at the end of the article and also available as an electronic file.

✉ Yeping Li
yepingli@tamu.edu

Extended author information available on the last page of the article

how existing research supports this notion with evidence of mutual benefits between design and STEM education.

In this editorial we make another extension of our previous discussion about the conception of thinking as plural proposed in our first joint editorial (Li et al. 2019b), with a focus on computational thinking (CT). Specifically, we take the position of viewing CT, as another example of *models of thinking,* which is important for every student to develop and apply in the twenty-first century. As there have been quite many studies and discussions about CT over the past decade (e.g., Denning 2009, 2017; diSessa 2018; Grover and Pea 2013; Wing 2006, 2014), we aim to build on existing research to provide a theoretical account of CT in this editorial and leave the discussion about educational programs and practices to develop students' CT as the topic for our next editorial.

In the following sections, we start by discussing motivation for conceptualizing CT and then propose a definition of CT that is applicable to STEM education and beyond. To clarify our definition, we further provide an overview of three primary approaches to describing CT in the literature, arguing that this body of literature has conceptualized CT as a stance toward programming competence and skill acquisitions, a cognitive process, and a particular type of literacy. We discuss how our definition connects with each of these three approaches. We conclude by describing how CT is distinct from other models of thinking like design thinking. We also highlight the implications of this definition for STEM education and future CT research.

Motivation for Conceptualizing CT Applicable to STEM Education and beyond

Wing's succinct article (2006) about CT has raised significant interest in professional and education communities (e.g., CSTA & ISTE 2011; Grover and Pea 2013). Wing argued that CT "represents a universally applicable attitude and skill set everyone, not just computer scientists, would be eager to learn and use" (Wing 2006, p. 33). This proposition reflects a rapidly growing interest in knowing, learning, and using computation in a broad array of professional activities in diverse fields such as physics, biology, and finance. Indeed, a recent report *Charting a Course for Success: America's Strategy for STEM Education,* released by the White House (December 2018), highlights that building computational literacy is one of the four pathways to succeed in STEM education, with one of three objectives under this pathway being "make computational thinking an integral element of all education" (Committee on STEM Education 2018, p. 23).

While the importance of CT has been commonly recognized, the meaning of the concept is contested. For example, Wing came to the concept from deeply inside the culture and technicalities of professional computer science. Someone outside that community might be prone to narrowly construe the idea of CT to direct connections with "number computation" or "computer." The following are example interpretations.

Viewing CT as Related to Computation Skill Development in School Mathematics

Computation is a familiar idea to most, especially to parents and students in elementary school. Students are required to learn to compute with numbers (e.g., CCSSI 2010;

NRC 2002). Such skill is commonly acknowledged as important not only in people's daily life, but also in preparation for, and in the conduct of, many professions, including science, engineering, insurance, and finance - wherever numbers are relevant. Computation is typically taken to be a basic skill, and parents and the public would be upset if children don't gain such basic skills through school education (e.g., Kakaes 2012; Kline 1973).

Computation was loosely connected to thinking until mathematics educators started to emphasize the importance of students *making sense* of what they do when they engage in computation (e.g., Brownell 1945; Li and Schoenfeld 2019). For example, students might simply memorize the subtraction algorithm and know how to compute, say, 45–21, as "subtracting a small digit from a larger digit," getting the correct answer, 24. But, without the needed *understanding* of place value and base-10 number composition and decomposition, then, a student might well carry out the computation 41–25 but get the same answer of 24. Here, we highlight the words of "making sense" and "understanding," as they require thinking beyond rote computation. Helping students to develop such deeper understanding has long been advocated and emphasized (e.g., CCSSI 2010; NCTM 1989), and also practiced in school mathematics such as the use of "number talks" (e.g., Parrish 2011).

Thus, in combining "computation" and "thinking" in this restricted sense, CT won't be strange to mathematicians, mathematics educators, and teachers at all. CT would be then emphasizing the importance of thinking and understanding in and for doing computation. The notion of CT might well be readily accepted for its importance to every student in learning mathematics. And yet, mathematics educators already have other terms that convey similar meaning, such as "number sense" (e.g., Sowder 1992) and "symbol sense" (Arcavi 1994). But, then, why should this new term have any particular importance beyond other, older ones? Why should the importance of CT be advocated by computer scientists as important to everyone, when computation in mathematics is commonly taken as a basic skill?[1]

Viewing CT as Specifically for Computer Scientists or Merely Learning how to Use a Computer

Widespread association with computers or programming can easily lead people to perceive CT as specifically for computer science professionals. It would therefore be difficult for many to understand why CT is important to everyone. Certainly programming—at least in the way it is perceived from the work of professional programmers—is considered difficult and esoteric. Developing software for a computer's internal operations would then be important to professionals in computer science, but out of reach to many others. In the same way, abstraction and modeling with the use of CT in many professional fields beyond computer science would be seen as unimportant and of marginal concern for most people. For more details, see the discussion of "vocationalism" in the CT movement in diSessa (2018).

[1] While numerical computation may be a basic skill, it is not necessarily mechanical and routine, as it is often perceived. For example, Dowker's (1992) work on estimation showed that mathematicians are flexible – they don't follow algorithms and may not use the same techniques when estimating the same quantities at different times. But they always have a good sense of how solid the estimates are.

In terms of making the concept of CT more accessible and relevant to those who are outside of computer science, it should also be noted that CT is not simply about learning how to use computers or software (i.e., "computer literacy"). By analogy, just learning to drive a car does not mean that one develops "mechanical thinking."

A Proposed Definition of CT

An adequate understanding of the notion of CT is clearly needed if CT is to be seen as important to everyone and worthy of being taught and learned in widespread educational contexts. Wing (2006) asserted that CT "involves solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science." (p. 33). The description provides a broad scope for CT's relevance. Wing (2006) further highlighted its importance with different manifestations of CT and its uses in many other fields, such as statistics, biology, physics, and economics.

At the same time, what makes CT special in Wing's description is the implication of "… drawing on the concepts fundamental to computer science." Wing (2008) further specified two essences of CT: abstraction and automation. The specifications strengthened the linkages of CT with core and general competences involved in computing and computer science.

Emphasizing "computing" or "programming" in CT, we then can conclude that CT has not been highlighted in traditional school education, as course requirements in computer science or programming are minimal or completely lacking. Wing (2006) should be credited with a notion of CT that is future-oriented and important to everyone. Through highlighting direct associations between CT and "concepts fundamental to computer science," Wing contributed substantially to the on-going movement of computer science education for all in the United States (e.g., PITAC, 2005; White House 2017).

However, challenges remain for many teachers and education researchers who struggle to understand the meaning of CT, its assessment, and usefulness for everyone (Denning 2017). The accessibility and usefulness of the notion might well be undermined for many by the expectation of training in computer science as a pre-condition. In fact, though, computer science itself is no longer viewed as the study of phenomena surrounding computers, but, instead, it is the study of computational information processing, both natural and artificial (Denning 2005, 2007). At the same time, human thinking can also be characterized as specific models of information processing when performing various tasks (e.g., Anderson et al. 2004; Simon 1979). The connections between computing and human thinking in information processing suggest the possibility of taking the notion of CT to a more generalizable level.

Specifically, we want to view CT as a model of thinking that is more about thinking than computing. As computing is the study of natural and artificial information processing (Denning 2007), CT is about searching for ways of processing information that are always incrementally improvable in their efficiency, correctness, and elegance. The entailed improvement can call for the use of various strategies (including abstraction and modeling), practice, skill acquisition and improvement. Here, information can

take different formats, at different levels of abstraction, and thus appears as various representations that can be customized and used in different disciplines for problem solving, modeling, and system building. Just as everyone designs and can design, we believe that everyone processes information, and helping them to do that well is our job as teachers.

Although programming and coding can be part of CT, CT should not be restricted in computer science but is prevalent in diverse professional fields and in daily-life events. For example, computational modeling has been used to summarize and analyze data (as code in CT) in different ways to help predict on-going trends in the coronavirus crisis, in multiple countries. A vignette of such an analysis by Andrea diSessa is included as supplementary material (see Computational Literacy in the Time of COVID-19). The lack of accurate data or CT would prevent people from effectively monitoring and managing the crisis development to save lives. Without specific attention to the improvement of information processing efficiency and elegance, we may lose opportunities to nurture students' CT and develop skills that prepare them to grapple with global crises. It is imperative that school curricula and instruction integrate CT in students' subject content learning, not just in computer science and mathematics but also in other STEM disciplines and beyond.

To further clarify our position, we take a brief review of different approaches in describing CT and discuss how our perspective is associated with these approaches.

Approaches to CT in the Literature

In the following sub-sections, we examine three different approaches that have had tremendous influence on the development of CT in research and educational practice.

Discipline-Based Approaches

The discipline-based approaches in describing CT have a long history associated with the development of computer science and computation itself in general. Denning (2017) indicated that George Pólya's work on mathematical problem solving (e.g., Pólya 1945) that provided general heuristics for solving a wide range of problems, as discussed in our first joint editorial (Li et al. 2019b), can be viewed as a precursor to CT. diSessa (2018) also identified many similarities between Pólya's work and Wing's writings about CT (Wing 2006, 2014). Although computation was in existence long before the creation of computers, the development of computation has experienced tremendous changes over the years associated with the invention and use of computers. Denning (2007) summarized the revolution in three main stages: (1) computation as a tool for performing simple and well-structured tasks, such as solving equations and running simulations, together with the creation and use of the first electronic digital computers in the 1940s; (2) computation used not only as a tool but also a method for discovering new knowledge beginning in the 1980s; and (3) computation and information processing found in the deep structures of many different fields beginning in the 2000s, such as biology, physics, and business management. Abstraction and modeling are essential to computation and computing as they develop and use in many different fields.

Related to the development of computation, computational science, and computer science, the notion of CT has also evolved from "algorithmic thinking" in the 1950s and 1960s (a mental orientation toward looking for algorithms that can help convert some input to an output in problem solving), a way of doing science that develops and uses computational models (associated with the development of "computational science", distinct from computer science, beginning in the 1980s), and as one of several key practices for every computer scientist whereas computation itself as existing in nature is viewed as more fundamental than CT (Denning 2009).

The discipline-based approaches in describing CT as discussed above suggest that CT can be characterized as a way of thinking and doing —a method— or as a key practice, which needs to be developed through programming practices. What is consistent among discipline-based approaches is the emphasis on one's capability of designing a correct solution with efficiency and elegance, in computational steps, that might reply on years' experiences and programming capability. As a computer scientist well known for his work on programming language and algorithms, Aho (2011) indicated that "Mathematical abstractions called models of computation are at the heart of computation and computational thinking. Computation is a process that is defined in terms of an underlying model of computation and computational thinking is the thought processes involved in formulating problems so their solutions can be represented as computational steps and algorithms." (p. 7) Aho's characterization of CT is consistent with what Wing (2008, 2014) emphasized as a key of CT: abstraction.

The historical development of the notions of computation and CT led Denning (2017) to argue that CT, as proposed by Wing (2006), represents a new version that is not the same as the traditional version developed through the history. The basic difference is that the traditional CT would be developed through programming practices in the profession, and the new version of CT would rely on concept learning to produce programming ability. Thus, the usefulness of the new CT to everybody is unclear and remains to be empirically studied (Denning 2017).

At the same time, the historic development of computation and CT also suggest that CT should not be taken simply as equivalent to computer science. The notion of CT has been used so widely in many different fields including mathematics and science both in the past and present. The position we propose to view CT is consistent with discipline-based approaches, in the sense that we emphasize the need and importance of performance improvement in efficiency, correctness, and elegance. In the computing field, performance can then be manifested as formulating and solving problems as computational steps and algorithms, and improvement can be made through developing and testing different computational steps and algorithms.

Psychology-Based Approaches

Although some scholars also characterized CT as a thought process in discipline-based approaches albeit from a computing perspective (e.g., Aho 2011), the emphasis placed on thinking rather than computing represents a shift of focus in the conception.

The study of thinking has had its own long history, as we discussed in our first joint editorial (Li et al. 2019b), which has evolved from philosophical discussion to psychological studies in and across disciplinary domains.

One particular approach that revolutionized the study of problem solving in the 1950s and 1960s was to conceptualize information processing in the human mind and use the computer to simulate human problem solving performance (e.g., Newell and Simon 1972). The approach was powerful as it built empirically from psychological studies about various components of human cognition and then tested them through software development and simulations. Examples include the Elementary Perceiver and Memorizer (EPAM, Feigenbaum and Simon 1984), and adaptive control of thought-rational (ACT-R, Anderson et al. 2004; Anderson and Lebiere 1998). At the same time, the approach was restricted in the sense that it did not really conceptualize CT at all, but used computation as a tool to help with research about human cognition. As discussed in the previous editorial (Li et al. 2019b), Simon expanded the information processing model of "problem solving man" from *Human Problem Solving* (Newell and Simon 1972) to the notion of "thinking man" in the book *Models of Thought* (Simon 1979). Simon used "thinking man" as a prototype to conceptualize human thinking as information processing in and through various component elements that can and should be merged into a coherent whole (Simon 1979).

At that time, there was very little emphasis on the idea that information processes and computation actually exist in nature. His book *The Sciences of the Artificial* (Simon 1969) contributed to building a foundation for the development of artificial intelligence,[2] including models of how the mind works. Now we can learn also from biology research; information processes and computation exist in nature as information encoding and generation in and through DNA, with its distinct computational methods (see Denning 2007). Thus, it is better to surpass Simon's notion of conceptualizing human thinking as information processes and computation. What makes CT, as proposed in our position, distinct from general human thinking then resides in the dedication to performance improvement in efficiency, correctness, and elegance, as discussed above.

Education-Oriented Approaches

Many scholars have discussed the ambiguity associated with CT and tried to clarify its meaning (e.g., diSessa 2018; Grover and Pea 2013; Hu 2011). For example, Hu (2011) reviewed and discussed many different perceptions that people held about CT, including CT as related to the tension between empirical and theoretical investigations; as an ability to see, comprehend and devise systems and processes; as an aid to do mathematics computationally; or as a set of problem-solving skills and techniques for software engineers in programming. Taking it positively, people view CT as relevant to many different professional activities, especially in different fields of STEM. At the same time, these diverse perceptions suggest the importance of clarifying the meaning of CT as is appropriate and needed for education.

There are three main approaches in describing CT that aim to facilitate educational practice.

[2] With a focus on problem solving, George Pólya was well recognized in the field of artificial intelligence (AI) for his work in heuristic but AI had not made good use of Pólya's work. Newell (1981) examined and discussed possible reasons.

(1) One approach in education is to follow a description developed from discipline-based approaches as discussed above. For example, Grover and Pea (2013) reviewed relevant development in K-12 education associated with CT, from procedural thinking development through LOGO programming in the 1980s (Papert 1980) to recent movement of several professional societies and organizations aimed to develop students' CT such as, the Association for Computing Machinery (ACM), Computer Science Teachers Association (CSTA), and Google. Instead of discussing possible clarifications that may be needed, Grover and Pea focused on how others might have interpreted Wing's description of CT. The review provided a nice summary of on-going efforts from professional organizations and studies that aim to develop CT through programming and computer science in K-12 education. Grover and Pea also highlighted and discussed that tremendous efforts are needed in developing programs and research further in a broad range of topic areas including curriculum, instruction, assessment, and teacher education.

(2) Another approach is to propose and discuss possible expansion of CT beyond computer science. For example, a recent report *Charting a Course for Success: America's Strategy for STEM Education,* released by the White House in December 2018, included CT, digital literacy, and computational literacy when laying out the vision for the United States to success in STEM education (Committee on STEM Education 2018). With the increasing use and importance of digital devices and the internet, the committee envisioned the importance of "digital literacy" as a basic level of understanding and "computational literacy" as a higher level of skill for all students to benefit from what technology development can bring for tomorrow's job opportunities. Building computational literacy was taken as one of the four pathways to success in STEM education, with "make computational thinking an integral element of all education" listed as one of three objectives under this pathway. Although possible relationships among computational literacy, CT, and digital literacy were not explained in the report, computational literacy was seemingly taken as having a broader scope than CT. The meaning of CT was explained first in the report with the definition from Wing (2014), and then expanded as including some broadly valuable thinking skills beyond computer science: evaluating information, breaking down a problem, and developing a solution through the use of data and logic. In fact, this expanded description shares many similarities with George Pólya's work on problem solving. With this expansion, the report further indicated the importance of developing students' CT as an integral part of all education with or without the use of a computer.

(3) There is one other approach that aims to highlight the importance of computation for students' learning beyond programming. diSessa (2000) advocated the notion of computational literacy before Wing's promotion of computational thinking, and also took a principled approach in emphasizing both "cognitive" and "social" aspects rather just focusing on programming and the computer environment.

Different from the popular use of literacy that many may perceive as "a casual acquaintance with …," diSessa (2018) defined literacy as a massive intellectual accomplishment of a culture together with a grand "re-mediation," shifting and expanding the fundamental forms of representation in society to include computation as universally known and used. Similar to the way algebra and calculus

> transformed the study of physics from a philosophical inquiry to a rigorous, precise empirical pursuit (diSessa 2000), computation not only supports many different fields (not just computer science) but can also change the very intellectual landscape of fields in what they do and how they develop. diSessa (2018) thus viewed computational literacy as important to every student, but not in the same way as Wing (2006). In his view, computation is not the special province of computer scientists, and everyone does not need to think like a computer scientist. Instead, computation is a fundamental resource for all of society, and it will develop earmarks that distinguish it in the way it is useful in each discipline and also in the larger, public society.

What diSessa (2000, 2018) advocated as computational literacy for everyone shares much with our position about CT in terms of the universal importance of computation and the emphasis on cognition.[3] At the same time, they differ not only in their approaches to formulating definitions for these two connected and complementary concepts, but in the patterns of appropriation of CT or computational literacy in the broader society. As such these different-but-related concepts also suggest different strategies for anyone anxious to get the most from computation in education and in the intellectual performance of society broadly.[4]

Differentiating CT from Other Models of Thinking

At the beginning, we indicated that we take the position of viewing CT, as another example of *models of thinking,* as being important for every student to develop and have in the twenty-first century. After discussing what we mean by CT, we need to explain further how CT may differ from other models of thinking. So far, we only discussed design thinking as another model of thinking in our second joint editorial (Li et al. 2019a). Thus, we would like to share some of our thinking behind identifying and defining specific models of thinking.

There are several principles that have guided our thinking: trans-disciplinary, purpose, and function. Taking CT as an example, if perceiving what makes CT special is the indication of "… by drawing on the concepts fundamental to computer science" (Wing 2006), people can wonder whether CT is pertinent only to computer science professionals or whether replacing the phrase of "computer science" with "physics", "life science", or "earth science", we can have "physical thinking", "life (science) thinking", or "earth (science) thinking" when solving problems. If CT is important to everyone, should physical thinking, life (science) thinking, and earth (science) thinking be all important to everyone especially when we all live and stay on this planet? The discipline-based thinking can be important but often carry limitations. Thus, CT, as a model of thinking in STEM education and beyond, needs to be conceptualized as truly trans-disciplinary and important to everyone.

[3] The vignette in the supplement provided by Andrea diSessa can serve as a good prompt for thinking about computational literacy and its relation with CT.

[4] For example, CT in our definition emphasizes thinking, per se. Computational literacy also emphasizes developing computational environments that are widespread, very easy to learn, and which have affordances for many, many uses, not just for discipline-specific ones.

Across different models of thinking in our perspective (see Li et al. 2019b), their differentiations can be made in terms of the purpose and function. For example, while design thinking focuses on designing and making things like everyone does and not just in engineering design, CT focuses on the performance improvement in efficiency and elegance. At the same time, these models of thinking can and should work together in various problem solving activities either individually or collaboratively in groups. Different forms of representations and abstractions can also be taken and used in different fields of study when these models of thinking may function for specific aspects of cognition in activities.

Coda

It becomes clear and important to us, in school education, to take an alternate perspective on CT and not restrict it to an association with programming or computer science professionals. While the practice of, and capability for, programming are certainly important for developing CT for computer science professionals, it is more important to realize that computation is an integral part of many other fields beyond computer science. The notion of CT should not be restricted definitively to computer science or programming, thus avoiding a *subject fixation* about CT. CT needs to be re-conceptualized, as we did in this editorial, to ensure it is relevant, important, and accessible to everyone. The development of CT can then be truly integrated into all education for everyone to succeed in STEM education (Committee on STEM Education 2018).

It is also important to point out that the reconceptualization of CT, as a model of thinking, makes its integration in all education a possibility. Great challenges remain to develop educational programs and practices to make the integration happen and to conduct research for further scholarship development (e.g. Honey et al. 2014). There is a rapidly growing number of programs and studies that focused on how CT can be developed through programming and computer science in K-12 education (e.g., Barth-Cohen et al. 2018; Bienkowski et al. 2015; Grover and Pea 2013), in and through STEM education with mutual benefits for students' subject content learning (e.g., Dauer et al. 2019; Sengupta et al. 2013; Yadav et al. 2018). In our next editorial, we will further discuss educational programs and studies to develop students' CT, conceptualized in different approaches. At the same time, we would like to take this opportunity to let everyone know that this journal encourages submission of related research on CT, its development in and through STEM education, through different theoretical lens and/or with the use of different research methodologies. It is a frontier topic in STEM education that calls for the development of new and robust scholarship (Li 2018).

Acknowledgments We would like to thank Christian Dieter Schunn, Eric B. Snow, and Pratim Sengupta for their valuable feedback on an earlier version of this editorial.

Appendix

Computational Literacy in the Time of COVID-19

Andrea A. diSessa
Graduate School of Education
University of California at Berkeley

Introduction: My Life as a Computationally Literate Person

I am among the privileged few for whom exercising computational literacy is an everyday affair. For example, I keep all my financial records in a self-constructed database, and I do financial planning with tools I've developed for myself. One of the advantages is that I get exactly the information I want, in the visible form I want, and I understand the details of how the tools work—for example, what assumptions they make, which is seldom or never true with on-line "calculators" such as those that estimate your income tax or plot a graph of future savings for retirement. I rarely use algebra to solve everyday mathematical problems because it's so much easier just to write a program. I prefer to write a program to compute, say, compound interest than to use a formula because (1) it's much faster than looking up or re-deriving the formula, (2) I can decide whether I want just a single result, a chart of gain over time, or a graph, and (3) I don't need to make any simplifying assumptions, such as a constant rate of return. To exemplify the last, in planning financially for our sons' college I wrote a little program that included both of our sons' expected tuition and board (and relevant dates), our estimated savings rate, raises in my wife's and my salaries, and also expected large purchases, such as a new car.

Professionally, I keep and analyze video data with my own tools. I outline papers in the same environment in which I program because that system has an excellent hierarchical organization facility (see images later), which is well-suited for good planning and editing, organized in multiple levels. I keep notes at meetings in the same way, since it produces a much better organized summary than linear text. And so on.

Some of this may sound sophisticated, but I am not a programmer. And many of these things are trivial, given a little knowledge of programming and a good environment in which to work.[5] In one of our classroom experiments, sixth grade students wrote programs that were far larger and more complex than what I normally make and use.[6]

But I'm sure my life is not like yours. What follows might be more vivid in your own experience.

A Vignette: Tracking COVID Infections

Here is an exercise that I did for myself, just out of curiosity. Well, curiosity and perhaps some fear of what COVID-19 meant for us and our communities. I wanted to track some critical inflection points on the progression of the pandemic. In particular, I wanted to track whether and when social distancing had any noticeable effect. Technically, I wanted to see if and when there might be a deviation from the ordinary, purely exponential increase one gets

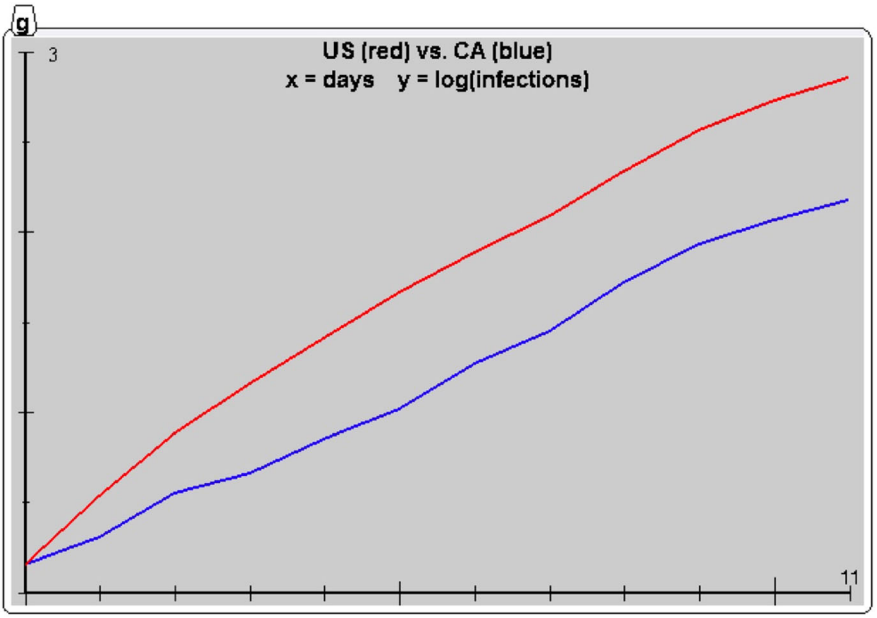
[5] I use the Boxer programming environment. We designed Boxer precisely to be a medium supporting computational literacy See diSessa (2000).

[6] They wrote video games. You can imagine how complex that might be, with multiple levels, scoring, a lot of narrative, different internal subgames, etc. Some of this is reported in diSessa (2000).

from a situation where each infected person infects a fixed number of other people. With social distancing, that number should decrease.[7]

It took me maybe 20 minutes to assemble what I needed, including finding an old graphing utility I had laying around, and also writing some new but simple utility functions. I keep mentioning speed because: (1) People who don't program don't know how easily such things can be done. (2) I would not do these things with programming if I knew a faster way to do them. I have no ideological commitment to programming just because it can be done. I do it if and when it's the fastest, easiest way I know to solve the problems I have.

I'll cut to the chase, and then backfill details. My very first graph, aimed to compare infection rates in California, compared to the US as a whole, appears below. The graph starts on March 18—two days after social distancing was instituted—and continues until March 29 of 2020.



This is a plot of log values of the data, since exponentials (which is the form of unfettered, constant infection rate growth) then appear as straight lines. The key lesson here is that, at the beginning of this time period, the slope of CA infections is significantly less than the US as a whole. That's what I expected to see. Selfishly, it's what I hoped for. To avoid confusion, note that I scaled the CA data so that it precisely matched the US at the beginning, just so that I could easily compare slope, there. (Rescaling and then applying the log function just provides a constant vertical shift; log ay = log a + log y.)

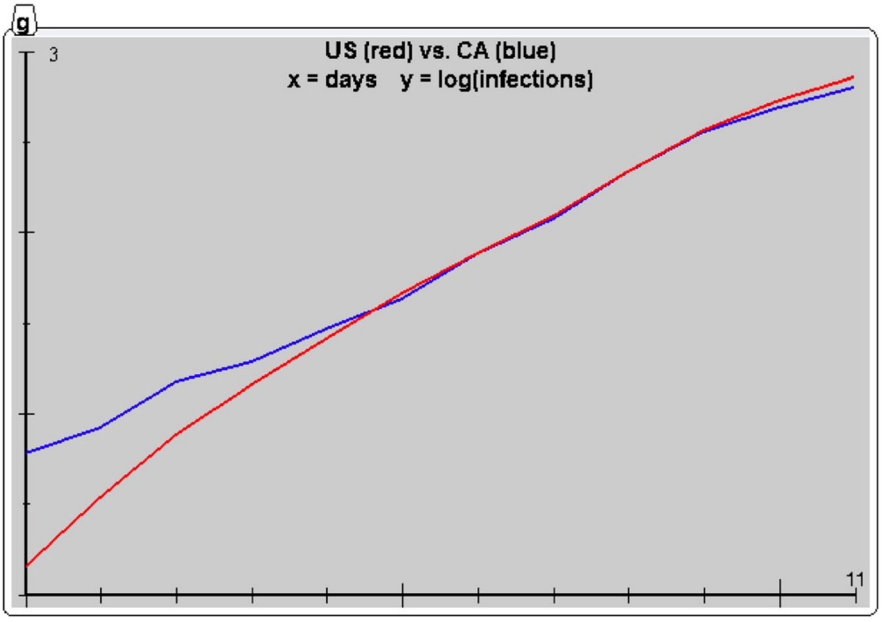
But there are interesting complications. While the blue (CA) graph is fairly straight, with the possible exception of a small downturn in the final two days, a colleague pointed out that it is a little jagged. It turns out that the jagginess is too much to be random error (1 / root N). But, a day later I read in the paper that California had been having trouble collecting data. Apparently, some counties were not reporting, or not reporting in a form that could be imported into the state's data

[7] Down the road, I expect to look for another inflection point, where the number of cases ceases increasing.

bases. That fact may resolve the puzzle of jagginess. But, it's also true that the graph is as good as the data. I used: https://ncov2019.live. Finally, I could not get to the site at exactly the same time of day each day, and the data was continuously updating.

Another observation is the arching curve in the US data. I still don't know why that is so. But, then, that was not an interesting point for me.

You can probably see that toward later times, the two graphs appear to be more parallel, signaling a similar infection rate. In order to make this clearer, I just re-scaled the CA data so that it matched the US data later in the graph.



In this form, it is clear that for a while the US and CA had the same slope (infection rate). But now, although both CA and US appear to have tilted slightly lower (smaller infection rates) in just the last couple of days, it appears that CA is *beginning* to show a lower, better rate. This is what I was hoping to see, even if I expected more than this little change. On the other hand, a change in infection rate should show up after about two weeks, so we are *just* edging into that regime. Yesterday, the day after I noticed the slight downturn in infection rates, I saw an article in the *San Francisco Chronicle*, entitled "Coronavirus slowing in Bay Area? Experts track data to see whether shelter in place is working." The article said:

> By day's end Monday [tomorrow], most of the Bay Area will have been holed up in their homes for two weeks — long enough, experts say, to see whether the unprecedented efforts to keep people apart are beginning to halt, or at least slow down, the coronavirus.

Yes, but you don't have to be an expert, if you're computationally literate! And I am learning so much more about COVID-19 and its tracking by doing it myself. Furthermore, I could not find any online tools to do what I wanted, much less the particular graphs I wanted. I could not even find any historical data listing, so that I have had to enter each day's data by

hand and keep my own historical dataset. These failures of the on-line world to give me what I wanted are cultural failures of our society with respect to supporting widespread computational literacy. Very few expect the public, now, to have any use for bare data, nor certainly the capability to do their own analysis of it.

What might my vignette have to do with education, other than suggesting that we should work to develop a more computationally literate public? I told someone that I could easily imagine working with a group of high school students developing hypotheses, tools, and analytical techniques like this—and in real time as the pandemic develops. I'm missing a wonderful opportunity (for not having a high school class to work with, just now). I said I was sure high schoolers could develop original hypotheses and appropriate analytical methods.

But, then, it occurred to me that we had already done something very like this, in a more difficult (if less compelling) case! We asked students to take data on the heating/cooling of two objects, at different temperatures, in thermal contact. We looked at the graphs, thought about how and why that happened (it's exponential decay; instead of change being proportional to amount (infections), change in temperature is proportional to the *difference* of temperatures). We then collaboratively developed a program embodying their model. In diSessa (2017) you can read about how one group of high school students developed, on their own and with no instruction, a normative model of temperature equilibration. In diSessa (2008) you can read several cases of students building conceptual and computational models of fundamental scientific principles, including early versions of our temperature equilibration curriculum unit. Indeed, in a later edition of the same project, we *did* teach eight grade students (from a marginalized, immigrant population) how to think about exponential growth (in the form of spreading rumors; "each one tells two").

The Nitty Gritty of Programming

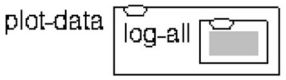
Finally, I want to further demystify the work I did building my little COVID exploration microworld. Just below is the database for California, as it exists today. The columns are, in the sequence specified by the **Key**: date, cases of infection, deaths, and the number of people who recovered. The "database" is just text typed or pasted into a "box." "X" represents missing data, all in the category of recovered cases, which I found for the US but never managed to find for CA.

Data Key
Date Cases Deaths Recovered

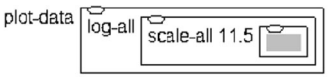
CA COVID
3-18 823 16 x
3-19 952 18 0
3-20 1219 23 x
3-21 1365 24 x
3-22 1642 30 x
3-23 1940 40 x
3-24 2494 50 x
3-25 2998 65 x
3-26 3909 80 x
3-27 4867 97 x
3-28 5549 119 x
3-29 6204 131 x

Next, I'll show the complete code for drawing one graph, revealed in stages. The first panel, below, shows the top level. Just plot a certain set of data points, which appears here as a black box.

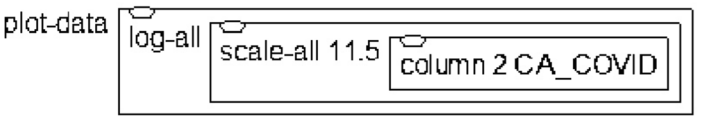
The next panel shows the black box opened (just click on it) to reveal that what's plotted are the log values of each element of another black box of data.



The third panel shows *that* black box opened up, revealing a data set consisting of yet another (black box) dataset, but scaled by a factor of 11.5. That happens to be exactly the factor that I needed to scale CA data in order to match with US data right at the beginning of my graphs.



Finally, with everything revealed, you can see that the input to the whole process is the second column of the CA COVID database, which is the number of reported infections.



That's the whole thing. The program to draw my COVID graphs is four commands in a nested sequence. **Plot-data** is a command the graphing utility understands. **Log-all** and **scale-all** are tiny programs I wrote to apply the named function to all the elements of a list of numbers. **Column** is an in-built primitive function of the system.

Coda

So, what's the world like when every citizen can program to the (very modest) level involved in this example? How will citizens then relate to the data-filled world in which they find themselves? What will schools be like? How will mathematics and science be taught differently? What different topics will be covered, how will basic conceptions of math and science change, and what different kinds of activities will students be engaged in—such as real-world data inquiries and modeling important scientific phenomena? That's computational literacy. You can read some of my own expectations and hopes in diSessa (2000) and diSessa (2018).

Acknowledgments

I thank Yeping Li, Geoff Saxe, and Melinda diSessa for helpful comments on earlier drafts.

References

diSessa, A. A. (2000). *Changing Minds: Computers, Learning, and Literacy*. Cambridge, MA: MIT Press.

diSessa, A. A. (2008). Can students re-invent fundamental scientific principles?: Evaluating the promise of new-media literacies. In T. Willoughby, & E. Wood (Eds.), *Children's learning in a digital world* (pp. 218-248). Oxford, UK: Blackwell Publishing.

diSessa, A. A. (2017). Conceptual change in a microcosm: Comparative analysis of a learning event. *Human Development*, *60*(1), 1-37. doi: 10.1159/000469693

diSessa, A. A. (2018). Computational literacy and "The Big Picture" concerning computers in mathematics education. *Mathematical Thinking and Learning*, *20*(1), 3-31. (Special issue on "Computational Thinking and Mathematics Learning.") doi: 10.1080/10986065.2018.1403544

References

Aho, A. V. (2011). Ubiquity symposium: Computation and computational thinking. *Ubiquity*, 2001(January). Available at http://ubiquity.acm.org/article.cfm?id=1922682. Accessed on 20 Feb 2020.

Anderson, J. R., & Lebiere, C. (1998). *The atomic components of thought*. Mahwah: Erlbaum.

Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., & Qin, Y. (2004). An integrated theory of the mind. *Psychological Review, 111*(4), 1036–1060.

Arcavi, A. (1994). Symbol sense: Informal sense-making in formal mathematics. *For the Learning of Mathematics, 14*(3), 24–35.

Barth-Cohen, L. A., Jiang, S., Shen, J., Chen, G., & Eltoukhy, M. (2018). Interpreting and navigating multiple representations for computational thinking in a robotics programing environment. *Journal for STEM Education Research, 1*(1), 119–147.

Bienkowski, M., Snow, E., Rutstein, D. W., & Grover, S. (2015). *Assessment design patterns for computational thinking practices in secondary computer science: A first look* (SRI technical report). Menlo Park, CA: SRI International. Available at http://pact.sri.com/resources.html. Accessed 28 March 2020.

Brownell, W. A. (1945). When is arithmetic meaningful? *The Journal of Educational Research, 38*(7), 481–498.

Committee on STEM Education, National Science & Technology Council, the White House (2018). *Charting a course for success: America's strategy for STEM education*. Washington, DC. Available at https://www.whitehouse.gov/wp-content/uploads/2018/12/STEM-Education-Strategic-Plan-2018.pdf Accessed on 18 Feb 2020.

Common Core State Standards Initiative (CCSSI). (2010). *Common core state standards for mathematics*. Available at http://www.corestandards.org/Math/Practice Accessed on 18 Feb 2020.

Computer Science Teachers Association, & International Society for Technology in Education (CSTA & ISTE) (2011). *Computational Thinking: Leadership Toolkit* (1st ed.) Retrieved from https://id.iste.org/docs/ct-documents/ct-leadershipt-toolkit.pdf?sfvrsn=4 Accessed on 8 Feb 2020.

Dauer, J. T., Bergan-Roller, H. E., King, G. P., Kjose, M., Galt, N. J., & Helikar, T. (2019). Changes in students' mental models from computational modeling of gene regulatory networks. *International Journal of STEM Education, 6*, 38. https://doi.org/10.1186/s40594-019-0193-0.

Denning, P. J. (2005). Is computer science science? *Communications of the ACM, 48*, 4 (Apr. 2005), 27–31.

Denning, P. J. (2007). Computing is a natural science. *Communications of the ACM, 50*, 7 (July 2007), 13–18.

Denning, P. J. (2009). The profession of IT beyond computational thinking. *Communications of the ACM, 52*, 28–30.

Denning, P. J. (2017). Remaining trouble spots with computational thinking. *Communications of the ACM, 60*(6), 33–39.

diSessa, A. A. (2000). *Changing minds: Computers, learning, and literacy*. Cambridge: MIT Press.

diSessa, A. A. (2018). Computational literacy and "the big picture" concerning computers in mathematics education. *Mathematical Thinking and Learning, 20*(1), 3–31. https://doi.org/10.1080/10986065.2018.1403544.

Dowker, A. (1992). Computational estimation strategies of professional mathematicians. *Journal for Research in Mathematics Education, 23*(1), 45–55.

Feigenbaum, E. A., & Simon, H. A. (1984). EPAM-like models of recognition and learning. *Cognitive Science, 8*, 305–366.

Grover, S., & Pea, R. (2013). Computational thinking in K-12: A review of the state of the field. *Educational Researcher, 42*(1), 38–43.

Honey, M., Pearson, G., & Schweingruber, H. (Eds.). (2014). *STEM integration in K-12 education: Status, prospects, and an agenda for research*. Washington, DC: National Academies Press.

Hu, C. (2011). Computational thinking – What it might mean and what we might do about it. *ITiCSE '11: Proceedings of the 16th annual joint conference on Innovation and technology in computer science education*, pages 223-227, https://people.cs.vt.edu/~kafura/CS6604/Papers/CT-What-It-Might-Mean.pdf

Kakaes, K. (2012). *Why Johnny can't add without a calculator*. Available at https://slate.com/technology/2012/06/math-learning-software-and-other-technology-are-hurting-education.html. Accessed on 20 March 2020.

Kline, M. (1973). *Why Johnny can't add: The failure of new math*. New York: St. Martin's.

Li, Y. (2018). Journal for STEM education research – Promoting the development of interdisciplinary research in STEM education. *Journal for STEM Education Research, 1*(1–2), 1–6. https://doi.org/10.1007/s41979-018-0009-z.

Li, Y., & Schoenfeld, A. H. (2019). Problematizing teaching and learning mathematics as "given" in STEM education. *International Journal of STEM Education, 6*, 44. https://doi.org/10.1186/s40594-019-0197-9.

Li, Y., Schoenfeld, A. H., diSessa, A. A., Grasser, A. C., Benson, L. C., English, L. D., & Duschl, R. A. (2019a). Design and design thinking in STEM education. *Journal for STEM Education Research, 2*(2). https://doi.org/10.1007/s41979-019-00020-z.

Li, Y., Schoenfeld, A. H., diSessa, A. A., Grasser, A. C., Benson, L. C., English, L. D., & Duschl, R. A. (2019b). On thinking and STEM education. *Journal for STEM Education Research, 2*(1), 1–13. https://doi.org/10.1007/s41979-019-00014-x.

National Council of Teachers of Mathematics (NCTM). (1989). *Curriculum and evaluation standards for school mathematics*. Reston: NCTM.

National Research Council (NRC). (2002). *Helping children learn mathematics*. Washington, DC: The National Academies Press. https://doi.org/10.17226/1043.

Newell, A. (1981). *The heuristic of George Pólya and its relation to artificial intelligence*. Pittsburgh: Carnegie-Mellon University, Dept. of Computer Science.

Newell, A., & Simon, H. A. (1972). *Human problem solving*. Englewood Cliffs: Prentice-Hall.

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. New York: Basic Books.

Parrish, S. D. (2011). Number talks build numerical reasoning. *Teaching Children Mathematics, 18*(3), 198–206.

Pólya, G. (1945). *How to solve it: A system of thinking which can help you solve any problem*. Princeton, NJ: Princeton University Press.

President's Information Technology Advisory Committee (PITAC) (2005). *Computational science: Ensuring America's competitiveness* (Report to the President, June 2005). Washington, DC: National Coordination Office for Information Technology Research and Development (NCO/IT R&D). Available at https://www.nitrd.gov/pitac/reports/20050609_computational/computational.pdf Accessed on 2 Feb 2020.

Sengupta, P., Kinnebrew, J., Basu, S., Biswas, G., & Clark, D. (2013). Integrating computational thinking with K-12 science education using agent-based modeling: A theoretical framework. *Education and Information Technologies, 18*, 351–380.

Simon, H. A. (1969). *The sciences of the artificial*. Cambridge: MIT Press.

Simon, H. A. (1979). *Models of thought. Volume I*. New Haven: Yale University Press.

Sowder, J. (1992). Estimation and number sense. In D. Grouws (Ed.), *Handbook for research on mathematics teaching and learning* (pp. 371–389). New York: MacMillan.

White House (2017). *President Trump signs memorandum for STEM education funding*. https://www.whitehouse.gov/articles/president-trump-signs-memorandum-stem-education-funding/ Accessed on 20 Feb 2020.

Wing, J. M. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33–36.

Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society A, 366*, 3717–3725. https://doi.org/10.1098/rsta.2008.0118.

Wing, J. M. (2014). *Computational thinking benefits society. 40th anniversary blog of social issues in computing*. Available at http://socialissues.cs.toronto.edu/index.html%3Fp=279.html accessed on 2 Feb 2020.

Yadav, A., Krist, C., Good, J., & Caeli, E. N. (2018). Computational thinking in elementary classrooms: Measuring teacher understanding of computational ideas for teaching science. *Computer Science Education, 28*(4), 371–400.

Affiliations

Yeping Li[1] · Alan H. Schoenfeld[2] · Andrea A. diSessa[2] · Arthur C. Graesser[3] · Lisa C. Benson[4] · Lyn D. English[5] · Richard A. Duschl[6]

Alan H. Schoenfeld
alans@berkeley.edu

Andrea A. diSessa
disessa@berkeley.edu

Arthur C. Graesser
graesser@memphis.edu

Lisa C. Benson
lbenson@clemson.edu

Lyn D. English
l.english@qut.edu.au

Richard A. Duschl
rduschl@smu.edu

[1] Texas A&M University, College Station, TX, USA

[2] University of California-Berkeley, Berkeley, CA, USA

[3] University of Memphis, Memphis, TN, USA

[4] Clemson University, Clemson, SC, USA

[5] Queensland University of Technology, Brisbane, Australia

[6] Southern Methodist University, Dallas, TX, USA