Title

Computational Literacy and “The Big Picture” Concerning Computers in Mathematics Education

Journal

Mathematical Thinking and Learning, 20(1)

ISSN

1098-6065

Author

diSessa, Andrea A

Publication Date

2018-01-02

DOI

10.1080/10986065.2018.1403544

Computational Literacy and "The Big Picture" Concerning Computers in Mathematics Education

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

8/18/17

Preprint: To be published in a special issue on "Computational Thinking and Mathematics Learning" in *Mathematical Thinking and Learning* (Vol. 20)

Abstract:
This article develops some ideas concerning the "big picture" of how using computers might fundamentally change learning, with an emphasis on mathematics (and, more generally, STEM education). I develop the big-picture model of *computation as a new literacy* in some detail and with concrete examples of sixth grade students learning the mathematics of motion. The principles that define computational literacy also serve as an analytical framework to examine competitive big pictures, and I use them to consider the plausibility, power, and limitations of other important contemporary trends in computationally centered education, notably *computational thinking* and *coding* as a social movement. While both of these trends have much to recommend them, my analysis uncovers some implausible assumptions and counterproductive elements of those trends. I close my essay with some more practical and action-oriented advice to mathematics educators on how best to orient to the long-term trajectory (big picture) of improving mathematics education with computation.

Introduction

The goal of this paper is to provide a "big picture"—succinct, but as broad and deep as I can manage in the space—of the best that we can hope for concerning computers in mathematics learning. I will be looking at historical and present trends, but the ultimate aim is to consider the future, in what directions progress would best be pursued, and what opportunities and pitfalls await us.

At the center of my analysis is one particular view of how we should construe the ultimate aims of engaging computers in STEM education. (I use "STEM" to emphasize the breadth of concerns when restriction to mathematics would be misleading.) That view has been gradually built and elaborated over the 45 years during which I have been thinking about and working toward improving STEM education with technology. It is a view that bears a strong family resemblance to that of many current workers in the field. In particular, I started my work in technology and education in Papert's Logo Group within the MIT Artificial Intelligence Laboratory, and the lines of work out of that profound beginning are strong and continuing (though hardly exclusive) in the range of work on "tech and ed." However, I will use my own formulation of the big picture in the interest of specificity and in order to raise some issues that I believe are important but not salient in other formulations.[1]

Putting forward one particular view might be interpreted as advocacy. But I am not aiming particularly to advocate, here. For *that* purpose, readers might consult my *Changing Minds* book (diSessa, 2000) (from which, however, I will borrow extensively). Instead, my aim is more analytic, to establish at least one clear and plausible reference view in some detail, and to pursue a family of issues that become salient in that view. I believe that many of these issues apply to many, most, or possibly all possible "big pictures" of computers in STEM education. A significant part of this paper will be making the case for how this view provides an analytical—sometimes critical—frame for considering other perspectives on technology and education, both close to and distant from my own.

The Literacy Model

My model of the best future for computers and STEM learning is easy to say. I view computation as, potentially, providing a new, deep, and profoundly influential literacy—computational literacy[2]—that will impact all STEM disciplines at their

---

[1] An excellent comparison to this essay from within Papert's contstructionist orientation is Wilensky and Papert (2010); both similarities and differences are illuminating.

[2] The plural, "literacies," is more apt. But I aim to keep the discussion as simple as possible.

very core, but most especially in terms of learning. In using the term "literacy" I need immediately to warn that I do not mean the popular, denatured use of "literacy" as "a casual acquaintance with …," which usage was particularly common near the beginnings of computer use in education. Instead, I mean to take on a serious comparison to one of the foundational achievements of modern civilization, mass literacy with the written word, which permeates not only all professional intellectual activity in STEM, but almost all learning and instruction in STEM. Indeed, developing textual literacy occupies a substantial proportion of all schooling.

Genuine literacies are big deals. Mass textual literacy took hundreds of years to achieve. There is a fairly extensive literature on textual literacy and how it works. So one need not start from scratch in understanding computational literacy. On the other hand, literacy theory is also a contested area that has gone through several major phases. It is also true that an absolutely critical set of issues for us—what *kind of* literacies can exist, and how do they come about?—has been severely neglected. So, on those matters, I have to make do, here, with some rough-and-ready ideas. In addition, what might be the same or different about computational literacy compared to textual literacy is not transparent, and it is also something that traditional theories of literacy do not touch. So, a literacy frame is helpful, but one has to extrapolate.

A definition and historical examples

Very roughly, I define a literacy as the adoption by a broad cultural group[3]—perhaps an entire civilization—of a particular infrastructural representational form for supporting intellectual activities. Two consequences follow. First, a literacy depends deeply on the particular representational form, text, for example, or computer programs (in a very general sense). What can be represented in the form? What is possibly represented, but difficult or marginally accomplished? What is impossible to represent and might need complementary forms? I call these questions—which concern the specificity of representations with respect to what they can represent—*the representation effect*. It is fundamental, and I elaborate it in several ways throughout this essay.

The second consequence of that definition is that the emergence of a literacy is a complex, extended social and cultural matter. In my view, nobody can plan a

[3] Literacy as it applies to written language might suggest that we should only consider civilization-wide candidates for new literacies. However, considering subgroups can also be important. Universal literacies might bud in subgroups, and, in any case, some subgroups might have very substantial impact on "whole civilizations." The technological-scientific community might substantially determine whether we survive as a species. For me, algebra is not in dispute as a literacy in this sense, and my guess is that calculus may well also be in the same category. See later discussion of both algebra and calculus as literacies. I believe computation will come close to universality, even if calculus, for example, is not.

literacy and enact it. So one has to look deeply at distributed social processes to understand the possible emergence of a computational literacy.

Textual literacy is so big, so complex, and so present, that it is analytically intractable. So I start with a boringly familiar—but effective—example of the intellectual consequences of new representational forms: the transition between Roman numerals and our current Arabic notation. Imagine multiplying CMLXXVII by, say II. Or by itself. What a mess! On the other hand, this task should be easy for upper elementary school students using Arabic notation, starting "7 times 2 is 14; so, 4; carry the 1." You might be able to do 1977 x 2 in your head, imaging the spatial layout as an organizing framework.

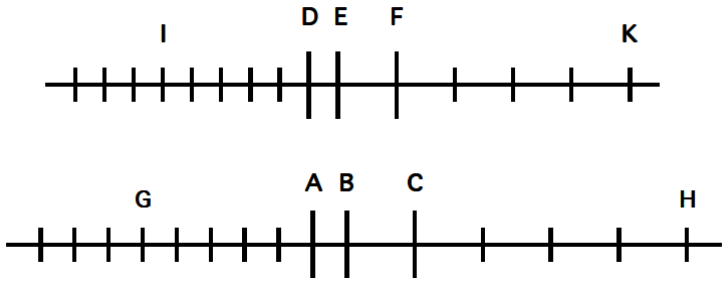
All of a sudden, a class of tasks becomes much easier. Social consequences can be enormous. Every shopkeeper can "now" do their sums and make change; a professional class, the calculational equivalent of scribes, which could be hired only by royalty, either expires or transforms into a different professional class that handles those tasks that are particularly complicated or specialized: such as financial advisors or tax accountants.

The first observation about this example concerns the representation effect. Every representation has its own affordances. It may make some things dramatically easier, but it may also have blind spots, things that are inconvenient to the point of uselessness. This requires consideration of the specific representation at issue and of the "fields of use" for that representation, about which I will have much more to say. Text may seem ubiquitously useful, but it is not. Calculation is narrower in use, but where it fits, it is more powerful. The second observation elaborates the fact that social consequences are critical. Before commerce, perhaps nobody in the civilization might care about the potential of Arabic arithmetic. See Saxe (2014) for vivid images of how certain cultures are completely content with minimal and enacted (rather than external and stable) representations of number, and how "silly little things," like commerce, can drive, slowly and in a complex manner, representational shifts in the culture.

I need another example to get to a reasonable social scale and a literacy more comparable to a possible computational one. For this, I turn to an experience I had, (recounted in more detail in diSessa, 2000). I have long been a fan of Galileo and his wonderful scientific accomplishments, and also a fan of his wonderful "naïve-conception-focused" expository mode: "Someone might think…, but imagine…." However, a long passage from one of his books completely baffled me for years. In it, Galileo proves six complex theorems about uniform motion, where distance is accumulated in proportion to time. Figure 1 shows one of the six proofs.

Theorem: If a moving particle, carried uniformly at a constant speed, traverses two distances, the time intervals required are to each other in the ratio of these distances.

Proof: Let a particle move uniformly with constant speed through two distances AB, BC, and let the time required to traverse AB be represented by DE; the time required to traverse BC, by EF; then I say that the distance AB is to the distance BC as the time DE is to the time EF. Let the distances and times be extended on both sides towards G, H and I, K; let AG be divided into any number whatever of spaces each equal to AB, and in like manner lay off in DI exactly the same number of time-intervals each equal to DE. Again lay off in CH any number whatever of distances each equal to BC; and in FK exactly the same number of time intervals each equal to EF; then will the distance BG and the time EI be equal and arbitrary multiples of the distance BA and the time ED; and likewise the distance HB and the time KE are equal and arbitrary multiples of the distance CB and the time FE.



And since DE is the time required to traverse AB, the whole time EI will be required for the whole distance BG, and when the motion is uniform there will be in EI as many time intervals each equal to DE as there are distances in BG each equal to BA; and likewise it follows that KE represents the time required to traverse HB.

Since, however, the motion is uniform, it follows that if the distance GB is equal to the distance BH, then must also the time IE be equal to the time EK; and if GB is greater than BH, then also IE will be greater than EK; and if less, less. There are then four quantities, the first AB, the second BC, the third DE, and the fourth EF; the time IE and the distance GB are arbitrary multiples of the first and the third, namely of the distance AB and the time DE.

But it has been proved that both of these latter quantities are either equal to, greater than, or less than the time EK and the space BH, which are arbitrary multiples of the second and the fourth. Therefore, the first is to the second, namely the distance AB is to the distance BC, as the third is to the fourth, namely the time DE is to the time EF.
Q.E.D.

Figure 1. A Galilean proof. From *Dialogues Concerning Two New Sciences*, Galileo Galilei. Translated by H. Crew and A. de Salvio, Northwestern University, 1939.

The astute reader will recognize that the theorem is basically "*distance* equals *rate* times *time,*" but in proportional form for two motions: $\frac{d_1}{d_2} = \frac{r_1}{r_2} \times \frac{t_1}{t_2}$. All the other theorems are simple variations. So, why does Galileo turn this simple fact into six theorems, and why is he burdening us with six independent and complex proofs?

You should know, but might not, that algebra in its modern form simply did not exist in Galileo's time. Toward the end of his life, while Galileo was inscribing his great insights in text and drawings, Descartes, Vieta, and others were at the beginning stages of the long and complex process of creating the notation and, more importantly, the algebraic literacy in which most of us are immersed today. Any contemporary high school student should be able to accomplish Galileo's "difficult problem," stating and finding the "simple" consequences of $d = rt$, with ease.

I was baffled by Galileo's proofs because algebraic reasoning was so natural and simple to me. Indeed, the properties of representations and their cultural power are not commonly appreciated. You probably learned that algebra was an Arab invention (maybe you recognize the Persian, Muḥammad ibn Mūsā al-Khwārizmī, who lived in the eighth and ninth century). The conventional designation of "the origins of algebra" betrays, I think, a systematic lack of appreciation of the importance of representations. Early "algebra," before Vieta and Descartes, was entirely discursive—represented in words only—pronouncing sentences and describing "algebraic moves" in natural language. Descartes himself seemed to deny the power of representational contributions made earlier by Vieta. My friends in mathematics at university glibly mocked Einstein's pride in a notational invention concerning tensor analysis; they deprecated it as "notation theory." In the larger scheme of things, even after Descartes' work, algebraic notation evolved significantly, and, most relevant to the present project, the long social transformations that took algebra from an exotic professional concern to a socially widespread literacy had not even begun.

On the social transformation of algebra into a true literacy, I make two points. First, social contest has been evident in that transformation pretty much to the present day. In the early 1900s, there was a prominent rebellion by leading mathematics educators against teaching algebra in high school. The teaching of algebra in secondary school declined from near 60% of students in the first decade of the 20th century to the mid-twenty percents in the 1950s (Klein, 2003). Modern percentages are back up and beyond those of the early 20th century, near 80 percent or above (Dalton, Ingels, & Fritch, 2016).

Not incidentally, rates for different demographics tell a critical tale of social contest and consequences. While rates of taking algebra are similarly high for three representative demographic categories (Whites, Blacks, and Asians) in the U.S., calculus tells another story. The most recent rates that I could find (for the high school class entering in 2009; in Dalton, Ingels, & Fritch, 2016; see also "Fewer Black Students Are Taking Calculus in High School," 2017) for enrollment in calculus for Whites is about 16%, but for Blacks, it was about 6%. While Bob Moses (and others, of course) seems to have succeeded by these simple statistical measures in making algebra a common "civil right," the same cannot be said for calculus. While the enrollment rates for Whites and Blacks in calculus are dispiriting, the enrollment rate for Asians tells a more positive story. According to

those same sources, the rate for Asians was about 43%. You will not be surprised to know that the rate for private schools ranked only a bit below that for Asians.

Apparently calculus has not become a widespread literacy,[4] and the demographics suggest a potentially serious problem with the distribution of the literacy to various groups.[5] Needless to say, dramatic problems with social distribution of textual literacy have also existed in all parts of the world, but perhaps most visibly in the vicious enforcement of illiteracy during times of slavery.

To summarize, while algebra has become a widespread literacy, with fairly massive implications for the present high tech and scientifically intense civilization, calculus has not. Social contest and critical societal issues, such as access to good, high-paying jobs, are deeply implicated.

Algebra/calculus literacy makes a good case study and landmark, all the more because it is a *technical literacy*. It is not much good for many cultural functions, for example, for poetry or courting. But it is useful enough, for enough important

---

[4] I am treating calculus as a potential literacy, sometimes nebulously distinct from a unified algebra/calculus. Here is how I see the relevant issues. To some extent, distinctions between closely related intellectual competencies are arbitrary. What we need is a more articulated view of the connections and disconnections between them, which would qualify *any* conventional choice of speaking about them as separate or unified. As for the specific relation between algebra and calculus, I recognize that calculus relies intimately on algebraic notation, with some potentially critical extensions (derivatives and integrals, obviously, but a lot more notational innovation appears when we get to vector or tensor calculus) that might be grounds, by themselves, to support an argument for "separate." On a larger plane, however, I think too narrow a focus on literal representations is counterproductive. Representations work in virtue of conceptual infrastructures, in addition to literal representational schemes. I believe the conceptual infrastructure of calculus is distinct enough to warrant separate consideration, at least some of the time. Or we would not appreciate what Newton had to do to develop his laws ("invent calculus"), distinct from the kinds of things that had been accomplished already using algebraic notation. So, in net, I am defending the sensibility of considering calculus to be, potentially at least, a separate literacy, while, at the same time, recognizing that it is not categorically distinct from algebra as a competence. The relationship between the "calculus of constraint" (currently "algebra") and the calculus of change (currently "calculus") might change with a computational infrastructure for learning.

[5] Of course, there are great complexities that I am not addressing. High school enrollment rates are not definitive measures for literacy rates. Maybe the racial/cultural gap is closed in college, at least for some students? What is actually covered in algebra courses? I simply have to suppress such important considerations here.

things, that our prospering and even surviving as a civilization (what is the change in rate of change of global warming?) are implicated.

With this in mind, I project that computational literacy will dramatically overshadow algebra/calculus literacy in the future. Computation is more generally useful, and it is far easier to learn. So it will be much more prevalent and important than algebra/calculus. I elaborate these themes, but for now I advance the proposition that, in its breadth and impact, computational literacy will wind up being roughly logarithmically halfway between the algebra/calculus technical literacy and, the grandparent and massively important literacy, competence with written text.

Allow me to start a list of literacy principles to help synthesize and summarize the discussion.

0. A literacy is a massive social/intellectual accomplishment of a culture or civilization, where many competing forces, over decades or centuries,[6] eventually settle on a particular representational form for wide-spread learning, use, and subsequent value. This is virtually my definition of a literacy, and a foundation for all the other principles.
1. Re-mediation: At the core of a literacy is the mass appropriation of a representational system, which shows distinctive and critical strengths, but also limitations and blind spots, and, thus, a possible complementarity with other forms of representation.

Reflecting on the Galileo example, I also assert that:

2. Literacies shift the basic intellectual structure of domains of knowledge along with learning trajectories and societal participation structures—who gets to do what. Concomitantly, *a literacy needs a literature*. One needs to transcend a representational system by itself, and get to civilizations' expanse of deep and powerful ideas. Algebra has permeated almost all scientific and technological literature. It is taken for granted, but its value is distributed among many profound and not-so-profound "things civilization has managed to think and explain" using the representation, constituting a profound and rich literature. This point expresses an anti-essentialist line, a literacy is not centered in any core power of the representational system, nor in special cognitive capacities that it allows.[7] I will pursue this idea later.

---

[6] Some believe that society is changing much faster these days. But if one looks at the massive intellectual changes that are entailed, it is reasonable to be skeptical that deep literacies can possibly develop quickly. We have already had four or five decades of widespread computation and not settled what that means for the education of our young.

[7] Several important historical accounts of literacy are, in my view, essentialist in this way. So, in this respect, I am pursuing a different line.

> Furthermore, the relevant "literature" is multifaceted. As the literature that uses algebra is centered in and at least somewhat distinct in different disciplines and foci, so will computation-relevant literature be. Write down the central algebraic principles in physics, economics, and statistics. Not only the literal expressions, but also the connected conceptual structures are diverse. As a physicist, for example, I instantly call up the relevant concepts pertinent to *particular* conceptual subfields from characteristic equations within them: classical mechanics, quantum mechanics, general relativity, and so on.

Galileo's conception of uniform motion was mediated by textual reasoning of a particular form, proof. However, his "complex" domain became unified and simple once it was re-mediated with algebra. Emblematically the domain became essentially a single intuitive equation, $d = rt$, along with some high-school-level inferences. "DRT" problems are simple exercises in modern algebra courses, exercises that don't even hint at the change in experienced complexity or in social distribution compared to Galileo's time.

Green Shoots of a Computational Literacy: The Mathematics of Motion

I would like to turn from big-picture aspirations to real-world examples of a budding computational literacy. About 25 years ago, my research group developed and taught a yearlong class for sixth grade children on the mathematics of motion. This turned out to be one of the most surprising and informative experiences of my professional career, especially regarding what is possible and what to expect even at early stages of computational literacy. One of the major elements of the class was student projects, notably creating computer games. While critical to what happened, I will initially (mostly) background this element while I follow a conceptual trajectory through the course using vignettes that connect to the general themes in this article. The trajectory might be characterized as *four easy steps to teaching vector calculus in elementary school*.

Galilean drop

This section provides a brief description of an early activity in our sixth grade motion class. It shows what can be done with almost no preparation, and it has proven to be a reliable way of scaffolding the "rediscovery" of one of Galileo's fundamental accomplishments: the description of "falling" as a motion with constant acceleration. In the following, I include some comments about our general experience using this activity on many occasions, in many contexts. In addition, the activity makes a nice expository bridge from the discussion on how algebra transformed Galileo's work to the following sections, which continue a fairly long intellectual trajectory of learning the kinematics of motion. For the fascinating details on the students' work in this activity, consult diSessa (2004; 2008). DiSessa

(1995) provides a more detailed theoretical analysis of how computational representations function to achieve surprising learning.

Soon after an initial two-week period of instructing programming, we asked our students to write a program to show "how a penny would fall if dropped from the Empire State Building" (which we simplified in future classes to the more everyday situation of a dropped ball). We provided a minimal and faulty initial model (Figure 2).

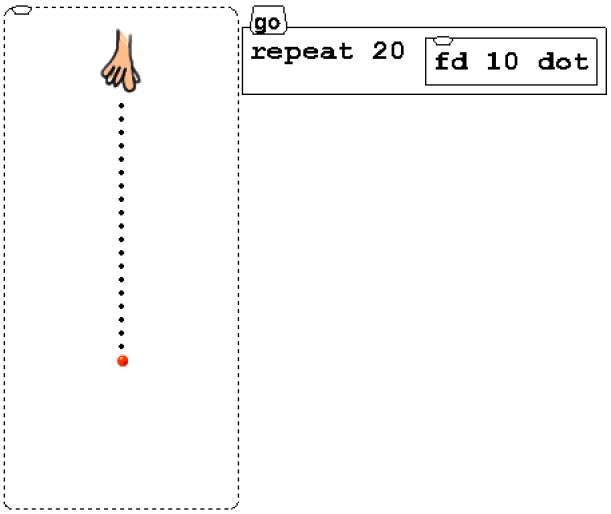


Figure 2. A starting model of an object's fall due to gravity. The program on the right produces the graphical output on the left.

The computer language used in the class is called Boxer,[8] and we designed it specifically as a medium suitable for experimenting with computational literacy. The notation should be relatively transparent, if unfamiliar. The model here says to repeat 20 times go forward a bit (10 units)—the instruction is literally "**forward**" (abbreviated as **fd**, which is set up to be in the downward direction)—and draw a **dot**.

Mixing small-group work and full-class discussion, students essentially always encountered a wonderful set of issues to debate, such as whether falling objects always speed up, and if so, how? How uniform ("even") is the motion, and what, actually, does uniformity entail? Many of these issues are the same ones that Galileo developed, himself, and, indeed, uniformity and simplicity were key criteria that Galileo used to prune possible models down to two, which are isomorphic with the two that almost always emerge as favorites in our motion classes (Fig. 3).

[8] I have to suppress the motivations and design principles for Boxer completely here. Consult diSessa (2000).

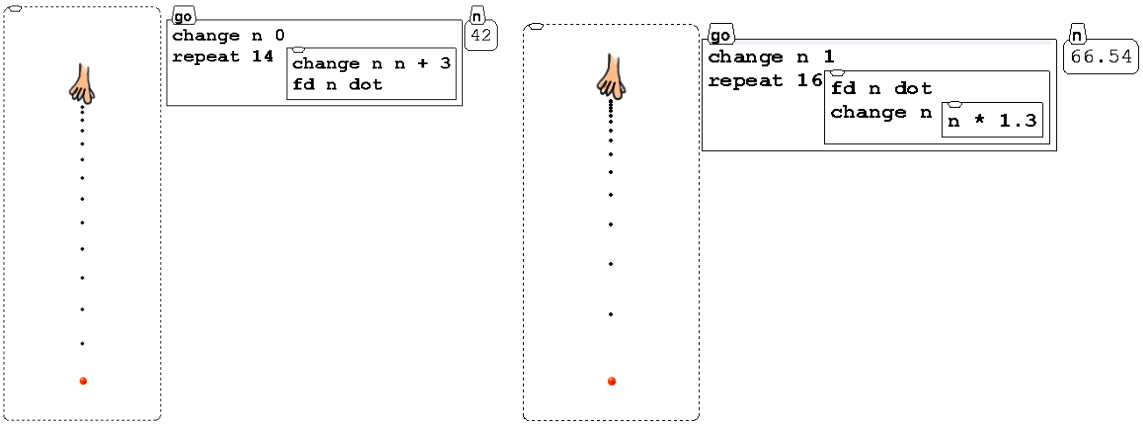


Figure 3. Two final candidate models (verbatim transcriptions of student work). To the square-cornered boxes (procedures), we now add round-cornered boxes, which are variables, such as **n**.

The model on the left is precisely Galileo's celebrated discovery: Every iteration ("increment of time") brings a constant increment to speed (represented by **n**). Galileo ruled out the model on the right with a sophisticated argument.[9] Students are incapable of understanding Galileo's argument (as high school algebra students would have no hope of understanding Galileo's proofs). So, for the final lap, we resorted to data, either from an ultrasonic motion sensor, or as measured in stroboscopic ("stop motion") images of a real fall.

To summarize, students as young as sixth grade, supported by a computational representational system, can profitably engage in something very much like one of the important intellectual enterprises in which Galileo also engaged. As algebra scaffolds a modern, easy conceptualization of Galileo's six theorems, computation scaffolds a fruitful inquiry into the nature of falling. Computational surrogates for velocity (**n**) and acceleration (the increment to **n**) arise naturally, along with more general and less technical, but central mathematical ideas, like "uniformity" as a good first guess, and often just the right idea.

[9] He used reductio ad absurdum: A proper submotion of the drop must take the same time as the full motion. In modern terms, he showed that the solution had to be (Dedekind) infinite, so of no relevance to the real world. A near-equivalent argument is that the multiplicative model cannot work starting from a zero speed. I do not recall whether we tried this argument with our sixth grade class.

The tick model

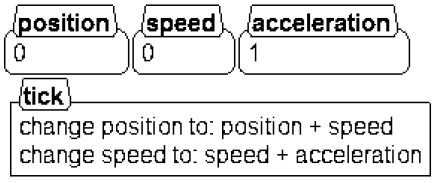


Figure 4. The TICK model.

Figure 4 shows what we call the "tick model" for motion. The metaphor is that, on each "tick of the clock," the **tick** actions are performed. In practice, the **tick** procedure is called inside a loop, which repeats it over and over (such as the **repeat** action in the drop models, above). In simple cases, that is all there is. In more complicated cases (games!), the loop contains other things, such as checking for object collisions. It is worth remarking (as we did with students) that the tick model is essentially the same as the students' version of Galileo's model, where **n** is identified as "speed" and a particular number (3 in Figure 3) is identified as "acceleration."

Tick *defines* the meanings of speed and acceleration, and it transparently shows how these quantities affect motion. **Speed** is precisely the increment to position during each tick, and **acceleration** is, similarly, the increment to **speed**. The command **"change position to: position + speed"** might easily be replaced or augmented with a graphical display of motion, such as **fd** (move a distance **speed** in the direction a graphical object is "facing"), as was the case in Galilean fall.

This model is more general than "distance = rate times time." If **acceleration** is not zero (whether it is constant or not), then **speed** (rate) is constantly changing. You don't need an equation for motion, such as $x = \frac{1}{2}t^2$; it just happens. Indeed, if acceleration or speed is adjusted in real time by the student, there *is* no equation that represents the motion. Arranging for real-time control is simple. One can, for example, introduce a slider control object linked to **acceleration**. Figure 5 shows such a slider as it appears in Boxer.

Figure 5. A graphical object, a slider, can be linked directly to the **acceleration** variable, which can then be changed using the mouse.

Conjoining the tick model with observing the motion that results pays rich conceptual dividends. Different speeds are visually apparent when **speed** is changed. Negative speed is a natural continuation of reducing speed "beyond" zero,

whereas, before the tick model, students denied that negative speed was at all sensible.

Similarly, one can see what different accelerations mean—not just the everyday meaning, "speeding up," but "acceleration" can mean slowing down, too, if velocity and acceleration have opposite signs. After a high school course and a term of university physics, students sometimes still mentally abbreviate "acceleration," as "speeding up," which has untoward conceptual consequences in physics. For example, they see a toss as two separate processes: decelerating up and accelerating down (Sherin, 2001). In contrast, a toss is just one continuous process to a physicist; the peak of the toss does not mark any change at all in the process itself.

Further experience with motion as controlled by the tick model takes steps toward understanding Newton's laws, $F = ma$. In particular, how does pushing and pulling control the motion of an object, in tick model terms? One can experiment with moving objects around using a controller to modify, in sequence, position, velocity, and acceleration. Little challenges develop understanding. For example, controlling velocity, how can you move the object from being stopped at one position, to being stopped at another? (You have to coordinate the return to zero slider value with the precise instant you are at your target position). Or, how can you do the same thing by controlling acceleration? Sixth grade students figure out how to do that in a few minutes. The ultimate question is how does one actually control motion in the real world? The answer is via acceleration, not directly via velocity or position (there is *only* an a in $F = ma$). While it cannot be obvious to students that that is how things work, experience with the tick model can prepare them for understanding the meaning and consequences of the fact.[10]

The tick model is easily accessible to sixth grade children. Mediated by computational representations, it is an approach to motion that has many advantages over an algebraically mediated approach:

1. Conceptual transparency: The meaning and function of speed and acceleration are easy to say and play out, even without literally running the program.
2. Phenomenological presence: The tick model inherently makes things move, unlike algebra. So, motion phenomena can directly interpret and enrich understanding of the model. The correlation between numerical phenomena (signs of numbers; relative magnitude) and perceptual phenomena ("turns around," "stops," etc.) becomes salient and explorable.

[10] One of the strong conjectures behind our course was that once a language of motion, provided by computational representations, was well in hand, Newton's laws of motion would be *much* easier to approach. Yes, this is true: diSessa (2008).

3. Generality: The tick model describes any motion, including motions that involve varying acceleration. Anticipating later learning, tick models also generalize in a different way to time-discrete models of the dynamics of any physical system, a mainstay of any modern study of complex systems, such as for weather or ecology. One just has to write down what happens "on each tick." Down the road, students will find discrete simulation of any system to be an obvious thing to do, and they will have skills for entering that practice.
4. Fun!: The tick model was a great mechanism at the core of many engaging individual and group projects. Students invented and implemented games using it. Many of the games and activities we designed and gave to the children had the tick model saliently and literally presented at their core. Ubiquitous contact is a wonderful learning principle.

Discrete calculus

In a continuous and algebraic world, the obvious representation for functions is with algebraic expressions. It is well known that, as motion and other mathematical topics are conventionally taught, many students get stuck believing that functions *are* equations (or just "expressions") and never get past that. However, this problem does not exist when motion is mediated computationally, rather than algebraically. In line with this observation, we chose to represent functions in our motion course as lists of numbers. You can look back at fall models (e.g., Figure 3) and imagine how easy it might be to convince students that a list of numbers is a great way to specify the changing speed of a motion. But now, a changing position is equally obviously represented by a list of coordinate positions. And a changing acceleration is just the same. One can ask (and we did) how do these "functions" (number lists) relate to one another? It shouldn't take but an instant for you to realize the relationship, and it doesn't take sixth grade students much longer. Figure 6 provides a prop for thinking. The "discovery" is that changes in position can be computed by adding up a sequence of speeds (which are literally small changes in position). Similarly, speeds are recoverable precisely in the difference between successive positions. The same relations exist between speed and acceleration. These relationships are the discrete version of the fundamental theorem of calculus.

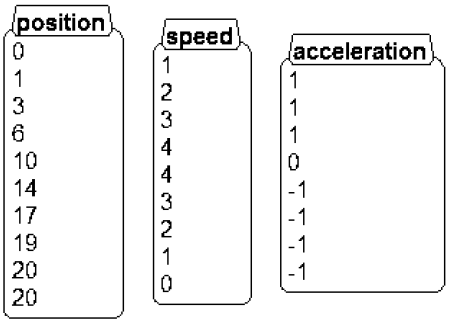


Figure 6. Position, speed, and acceleration as functions of time, represented as lists.

I want to call out several things. First, these relations were regarded as obvious to all the members of our sixth grade class. Indeed, one of the students spontaneously proposed that an excellent representation for a changing motion would be a list of speeds long before we introduced number lists in Boxer. And, he casually added that you could even add up the speeds to get distance. Second, creating and using number lists of positions or velocities is a trivial addition to tick models. Just insert a command to "**append** <a velocity or position> **to:** <a box>." So, the fundamental theorem of calculus can be easily operationalized, checked, and used. It has easy experiential and even practical presence (e.g., collecting data) in working with motion in a computational medium.

Again anticipating later learning, number lists and the discrete version of functions generalize better than, for example, the "slope of the curve" and "the area under the curve" to more advanced differential and integral forms. Algebra and calculus hide, rather than reveal, simple qualitative facts underlying the study of change. Adding up little bits that then gives information about boundary values (e.g., "change over an interval") generalizes much better to vectors (just below), but also to differential forms like divergence, curl, and vector field index theorems (e.g., Atiyah-Singer). We have, in fact, taught high school students the Gauss-Bonnet theorem (Abelson & diSessa, 1981), the Atiyah-Singer index theorem, and the whole class of Stokes-like theorems (Gauss' and Stokes' theorems as special cases) using the core idea of "adding up bits" and "cancelling at the boundary."

Vectors and the calculus of vector functions

Our first proposal for funding to the National Science Foundation was rejected because, they told us, the instructional program was absurdly ambitious. A core example used against us was that vectors are "developmentally inappropriate" for sixth graders. Everyone knows that vectors are difficult even for high school students, and one doesn't usually get any hint of vector calculus until university. We toned down the proposal, eliminated talk of vectors, and were funded. But, then, we taught vectors to our sixth grade students, and it wound up being an easy and resounding success. Here's how we did it.

The key goal was getting vectors into the computational representational system in a deep and functional way. Bruce Sherin implemented vectors for us in Boxer, and they looked like Figure 7.
.

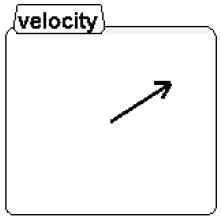


Figure 7. A vector in Boxer appears as a box with an arrow in it. The arrow tip can be moved around with the mouse, and there are commands to add vectors, multiply them by numbers, and **move** objects according to a vector specification—the vector equivalent of **forward**. Vectors can be unnamed or named (corresponding to "bare" numbers and variable numbers, respectively).

A keystroke makes a new vector, quicker than typing a multi-digit number. With a few keystrokes, one can create two vectors and add them by executing the command shown in Figure 8. You should imagine a student playing with changing the added vectors (mouse drag), and seeing what results from their addition (the result of a "do-it" key press).

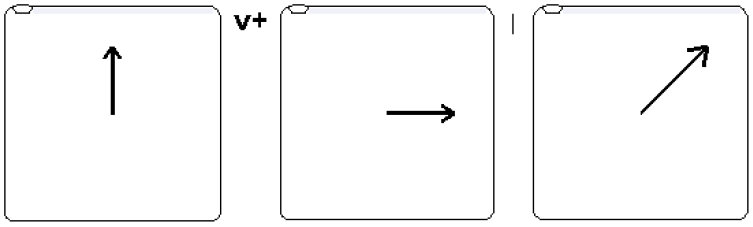


Figure 8. **V+** designates vector addition.[11] Pressing the "do-it" key (execute) results in a returned value, the right-most vector.

The core of a tick model suitable for experimenting with vector velocity and acceleration appears in Figure 9. Imagine dragging the acceleration around in real time in order to try to get an object to move in something like a circle. We had an "orbiting spaceship" game that presented just this problem. Students quickly got better than us at managing a rough circle (acceleration always toward the planet).

[11] To keep the exposition brief, I use a notation different from what our class used.

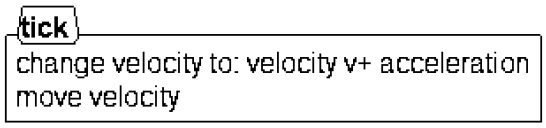


Figure 9. A tick procedure for experimenting, in real time, with how velocity or acceleration controls the motion of an object.

We prepared a fairly elaborate curriculum for teaching vectors, but the tasks we designed turned out all but superfluous. Vectors were such a hit with our students that they spent massive amounts of time playing with them, and, as with the linear tick model, building games using them. Vectors make a superb game interface, as well as a programming element. One student, for example, implemented a game where you, as the captain of a boat, are trying to land at a dock without crashing into it. A vector represented the direction and power applied by the boat's motor, which the boat driver/game player controlled. An internal variable representing the tide changed in a somewhat random way to make the game challenging.

All of the advantages that I mentioned for the tick model accrued to vectors. After considerable, often self-motivated, experience, they became conceptually transparent (e.g., what vector addition means; how vector velocity or acceleration relate to motion). The "considerable experience" involved simultaneous presence of the experiential aspect (motion) and the computational analytic (vectors as adjustable controllers of motion). The motions generated were liberated from those representable by simple algebraic expressions. And, vectors were regarded as fun in the students' freely chosen projects.

We asked students to replay the Galilean model of a drop, this time with an upward toss at an oblique angle. We presented them with a stroboscopic image of such a toss and then asked them to select a vector function of time (a list of acceleration vectors, each of which, of course, the students could modify with their mouse) in order to make an object re-trace the real (stroboscopically presented) motion. The students pretty quickly zeroed in on the surprising fact that constant acceleration, again, lies behind gravitational trajectories.

Review of principles

Let me review the literacy principles announced earlier as they relate to this case, starting with principle 1, re-mediation. Quite clearly, use of a computational medium massively influenced the experience students had in learning the mathematics of motion. Computation seems almost an ideal representational system in which to study motion, in substantial degree because programming is centrally about actions across time. A study comparing students' learning with algebra with learning with programming (Sherin, 2001) showed a distinct bias for the students using algebra to emphasize some things (conservation, expressed well in static, algebraic form) and to be quite poor at basic time-dependent behavior (as is far better represented in tick models). Programming students were much better at time-dependence.

In our motion class, computation re-organized the intellectual domain (principle 2) in profound ways. This is easiest to see in the learning sequences. Students, here, learned about motion: velocity and acceleration—even in vector form—and even a legitimate form of the fundamental theorem of calculus in mathematically precise ways (but not in ways that are familiar to most people) years before they would have the skills with algebraic representations and continuous functions to accomplish similar things if mediated by that "old" representational system. But, the shift to a computational representation also has corresponding deep epistemological shifts, such as essentially viewing our universe as discrete in time. With respect to the long social road to adopting a computational literacy (principle 0), I know that mathematicians and mathematics educators (and physicists, too) will trip over the epistemological shifts encountered here, and likely will resist them. I will treat some of this later, after some further preparation. However, based on our extensive experience, I am confident that, eventually, the computational mathematics of motion will be viewed, at minimum, as both legitimate and a very effective stepping stone to a "grown up" mathematics of motion.

Two more points about cultural change: First, the perennial complaint about learning the representational system, programming—"I have no time for anything else in the curriculum"—should be wilting. We spent two weeks teaching programming before we started in earnest on motion…out of a full year of instruction. Of course, that effort would easily be further amortized if other courses employed programming, as well. Even with two weeks of work, students complained that we spent too much time with programming. They maintained that it delayed the really fun and interesting parts. And, besides, they said, programming was better learned as-you-go, in doing the "fun" things. If I were to re-do that course now, I would heed the students' advice.

As our society gets more experienced with programming, the bugaboo that it is "hard and takes too much time" will fade. See also related societal trends in the next section. We also discovered that the most important parts of the learning that our students accomplished were often, like tick models, incredibly simple from a programming point of view. However, I am far less sanguine about how current curricular standards constrain the changes needed in instruction to enact computational learning like this. Or worse (again, see later discussion), learning programming will continue to be rigorously separated from core mathematics (or from STEM, in general).

I believe the tick model and our way of approaching calculus for sixth grade children should be regarded as part of a new, deep literature engaging a computational literacy (incorporated in principle 2). We should begin thinking about literatures differently than "a library full of books," and more like some key conceptual ideas (e.g., representing motion in programs) and associated activities and learning goals. However, I also don't expect landmark presentations and textbooks to disappear very soon.

Finally, I want to introduce two new principles that are salient in what happened with our motion class.

3. A computational literacy can transform—revitalize—the ecology of learning activities. Much of the power of our motion curriculum resided in the enthusiasm exhibited by our students toward it. Activities exercising the relation between motion and computational formalisms (tick models) were taken as good fun, and often an inspiration for personal variations. For example, one sixth grade girl thought that maybe there should be something "beyond" acceleration, and wrote a program involving a variable that incremented acceleration, which, in fact, is called "jerk" by physicists. It turns out that it is almost impossible to control an object using jerk, which this student discovered. More generally, the freedom to take these ideas and put them to personal use (concretized most clearly in their projects of designing and implementing motion games) gave an extraordinary spark of life to the class. For me, the comparison of what happened in this class with the way we teach algebra—one personally irrelevant and deadly dull exercise after another—is earthshaking. Here is where the fun our students had takes deep root in the development of a computational literacy.
4. Reformulation (cognitive simplicity). The prominent visibility of representations may distract from a concomitant and equally important thing that is going on here with motion. The discrete formulation of motion means that children can think about it using everyday conceptual strategies, for example, comparing categorical changes (e.g., "before and after"), and imagining sequences of these. In our motion course, we could see students using these strategies, but adapting them skillfully to assumptions of continuity. For example, they drew stepladder "graphs" of motion, but did not assume speed changed in big jumps. Instead, they interpolated between "before" and "after" appropriately (diSessa, Hammer, Sherin, & Kolpakowski, 1991). Newton's method of "blows" (he construed "forces" as a sequence of discrete "hits") shows the same cognitive simplification, but without the systematic framework of computational representations. Reformulation is too complex and separate a matter to pursue in depth, here, although I will drop some tidbits into the conversation later. But the upshot is that there are ways of approaching subject matter that work better for reasons other than (or in addition to) re-mediation. In general, my personal formulation of reformulation involves understanding the character and potential power of common, "naïve" ideas in scaffolding technical competence (A detailed exemplar is diSessa, 2017; a popular presentation is contained in diSessa, 2000).[12] When such ideas are engaged to show

[12] The currently popular pursuit of discovering "learning progressions" (Hmelo-Silver & Duncan, 2009) with good properties recognizes the essence of this point. Unfortunately, the related literature has simply not concerned itself with representational infrastructure, much less the development of literacies that may transform what our society regards as "easy" and "hard," and what is taught in

surprising cognitive simplicities *and* when they align with a powerful representational change, as for motion here, learning becomes amazingly transformed, faster, and easier.

Caveats and concerns, and the inherent implausibility of major reformulations and re-mediations

There are sensible questions about the legitimacy of this whole program of coming to understand the mathematics of change. For example, in what sense, really, do numbers in a **speed** variable represent speed, as opposed to merely small distances? When might it be necessary to teach the "real" (continuous) mathematics of change, as opposed to an "approximation" to it? Will the time-discrete version be a help or hindrance to that learning?

Some of these questions have relatively easy answers, even if they might stress contemporary epistemological assumptions about these matters.[13] Other questions, such as concerning long-term learning trajectories (e.g., to "real" calculus), will take more time to answer convincingly, although I feel I see the path quite clearly in exposing important, simple qualitative relations that are salient in the computational formulation, but hidden, nearly invisible, in algebraically-mediated views of motion and change. Initial resistance and long periods of incubation are undoubtedly the norm for substantial re-mediations and reformulations in anticipation of new literacies. The necessity of long-term cultural change, including some trial and error, may be one of the few certain things about them. But, we surely will not get there without trying, and considerable experience with some of these ideas shows very promising results.

---

school. Another prominent line of work on "misconceptions" and "naïve theories" regards pre-instructional knowledge as full of errors and blocks to learning, and minimizes—if it is considered at all—the pursuit of cognitive simplicity, *helpful* naïve ways of thinking. See diSessa (2017).

[13] There is no space in the text for convincing elaboration. But in this note, I touch on a few issues. Many people trip over the fact that our mathematics of motion is based on time-discrete models. Here's a playful, but to-the-point reply: Suppose that scientists have discovered that time is actually discrete, but at a scale something like 10^{-20} seconds. Then I challenge them to come up with a realistic way to observe that granularity. An even better view is that computation provides a perfectly fine model of motion. And all we have with regard to the world is better or worse models. Suitably deployed, computational/discrete models are both powerful and much more learnable—at least concerning some things. There is nothing more to ask. Finally, concerning whether a "small" distance can legitimately represent speed, I put forward the principle that the meaning of numbers lies primarily, if not exclusively, in how they are used. Scaling ("proper units") is a fine thing to learn, but it is separate.

The Development of a Computational Literacy: Current Trends

Thinking about cultural change

You would think that the development of literacies would attract a lot of study. But that is not the case. The literature on "the theory of literacy" deals almost exclusively with what literacies do and how they do it. Empirical study of the development of literacies seems minimal. I have struggled finding relevant study of the development of the algebra/calculus, at least study that makes contact with any generalizations about literacy development. The history of text-based literacy is pretty well documented, but nearly devoid of generalizations that can yield insight into *new* literacies that might be different from textual literacy in substantial ways (e.g., specialized to "technical" literacies, as computational literacy will be).

So, in preparation for examining contemporary trends, I will here present a "toy" model of literacy development that will, nonetheless, be useful in following sections.

Let us consider a culture to be a repository of a lot of different kinds of things that are historically developed, but also responsive to the contemporaneous environment. Here is a short list of such "cultural elements," which I'll dub MMVSS (pronounce as "Em Vees").

Movements – Cultural history is full of movements that galvanize many people in concerted efforts to change things in a certain way. Examples of political movements are a dime a dozen, but not "cheap": They can have world-shattering impacts. In education, per se, movements have been important (despite the amazing resistance of educational systems to change). For example, the anti-algebra movement in the early 20th century was one such, and, more recently, the algebra-for-everyone movement has, I think, had substantial and opposite impact. Constructivism has had a continuing great impact, as has "Back to Basics." Some of the movements relevant to the potential rise of computational literacy, however, may originate outside education, per se, or, more particularly, outside of educational research. Re-representation and, especially, reformulation have not been a substantially developed part of educational research, certainly not concerning a scale of change like that presented by computational literacy.

Memes – At the other end of the "size" spectrum, little but influential ideas can accumulate into great impact. One that will be relevant to later discussion is the idea that "nerd culture" can be both interesting and attractive: "Nerd Power." If you take to listing recent movies and television shows that celebrate the subculture of scientifically/technically brilliant people, who are most often depicted as a bit socially awkward, you won't soon run out of examples.

Values – What should we prize in an educational experience? The traditional focus on rigor and coverage of "the standard material" should be expanded to include far

deeper engagement, especially in view of modern societal conditions: dealing with diversity in schools and the need to make STEM learning more attractive to a wider range of students. If there is a trade-off in terms of societal good—which is not obvious, given the ad hoc and scientifically untested societal value of traditional goals and curriculum—then, still, we should tilt instruction in the direction of what my group did with our sixth grade students. No doubt this will be controversial and contested in the throes of the revolution toward a computational literacy.

Outside of school, Papert suggested that, in the way that ecology has become a deep value in many corners of contemporary society—everyone should do their part—so should "mathetics," the science of learning,[14] making as much of our civilization as possible attuned to children's learning. Whatever the status of this idea, there is no discernable movement, yet, comparable in any way to ecology.

Sensitivities – A principle part of modern "first-world" culture is highly sensitive to, for example, issues of equity and prejudice. Obviously, this was not true in the past, and we can hope that these issues will someday be enough in the "gears" of the working society (structural non-racism) that explicit attention may be less necessary. Anticipating the special concerns of computational literacies, the epistemological power of evolving new representational systems, including the very possibility of a new literacy, may not currently be particularly resonant with popular culture.

Sensibilities – How do and should things "work"? Who gets to say what should be taught in the mathematics curriculum and, even roughly how? Mathematicians? The developers of accountability measures or standards documents? Who was it that said, "vectors are completely age-inappropriate for sixth grade students?" A true computational literacy will without doubt disrupt current assumptions, and cultural ruptures will appear in the process.

I need one more idea to complete my toy theory of cultural change. The idea is resonance or synergy, where some changing MMVSSs come to mutually reinforce and blossom into larger and stronger trends. An upcoming example will be highly relevant to the emergence or delay (sidetrack, short-circuit) of a potential computational literacy.

I now turn to two trends in the use of computers in mathematics (STEM) education in view of the considerations above.

The first trend is a highly visible one, and one that has already been extremely consequential. It may not be salient to many mathematics educators, but that is part of the point.

---

[14] The term "mathetics" was coined by John Amos Comenius in the 17th century to mean "the science of learning." Papert also used the term in the same way, as a contrast to the science of instruction.

Computational thinking

Jeannette Wing deserves credit for raising the visibility of teaching *computational thinking* as a potentially very important concomitant to the rise to prominence of computers and computation in our civilization. Computational thinking shares some features with computational literacies. Indeed, Grover and Pea (2013), in a review of the research literature relevant to computational thinking, judged that, while computational literacy was a concept developed well before computational thinking, the phrase "computational thinking" was probably a better one. It is more modern sounding and stands less chance of being confused with "computer literacy," which movement I took pains earlier (as I always do) to distinguish from computational literacy. But, the problem in Grover and Pea's analysis is that computational literacy and computational thinking are quite distinct, even divergent, which they either did not notice or chose not to mark. (Although, to their credit, they do note that computational thinking has not addressed how its ideas come down to brass tacks in learning domains such as mathematics—which is a primary focus for computational literacy. See continuing discussion.) This is not an issue of choosing terms; it is an issue of choosing directions. Regardless of where one comes down in terms of advocacy, we should certainly have an analytical frame that can separate these two ideas, and other such ideas and movements. That is at the heart of this article, and it is front and center in this section.

Wing, in a series of articles (Wing, 2006; 2008; 2011; 2014), brought the idea of computational thinking to public prominence. Wing's definition starts heuristically with a rather parochial view, "thinking like a computer scientist" (in all of the cited articles). But she adds extension and precision. For example, "Computational thinking involves solving problems, designing systems, and understanding human behavior, *by drawing on the concepts fundamental to computer science*" (e.g., Wing, 2008, emphasis added). Her latest and most technical definition (2014) is, "Computational thinking is the thought processes involved in formulating a problem and expressing its solution(s) in such a way that a computer—human or machine—can effectively carry out."

Wing draws— "masterfully" might be appropriate—on a very wide range of facts and MMVSSs in contemporary culture. Here's a sample:

- Computers are ubiquitous and deeply embedded in our modern world. People ought to learn about them.
- "Coding," all by itself, is a modern and fun thing to learn. It ought to be made *more* fun and widespread.
- Equity: Computer science ideas, like all powerful, general intellectual capacities, should not be bottled up in computer science tracks, but should be made available to everyone.
- Many disciplines are being transformed by computation: statistics, by being able to deal with huge, real-world data sets; biology, by algorithms for gene sequencing; physics, by the need for detailed and specialized physics in the pursuit of quantum computers; etc.

- Computer scientists know that computation is powerful, beautiful, and engaging; everyone should experience that.
- But, perhaps the central-most leitmotiv of her writing is that computational ideas are *generally* powerful, applicable in principle by everyone to a very wide range of situations. Literally the first text of her first article is: "[Computational thinking] represents a *universally applicable attitude and skill set* everyone, not just computer scientists, would be eager to learn and use." (Wing, 2008, p. 33, emphasis added) A significant proportion of each of her articles deals with this particular meme.

The pervasiveness of this last point in Wing's writing makes it look as if she is advocating a kind of computational literacy. Perhaps. But it is not a computational literacy like the one toward which I am pointing. Many of the principles I mentioned differentiate these two directions, which I will make clear directly.

The second to last meme, above, has a strong resonance for computer scientists: Feelings for the power and beauty of their fields characterizes virtually all disciplines. Wing gave voice to computer scientists' feelings, but more importantly, enrolled them in an activist position in the whole computational thinking movement. Subcultures and subgroups can be incredibly important in movements, and their own values and memes, thereby, become important. I don't think it is a secret or a surprise that computer scientists have been strongly at the vanguard and in the trenches of the computational thinking movement.

Computational thinking has taken off and has had big influences. The wider social uptake involved a greater scope of MMVSSs than Wing's emphases. For example, "coding academies" that advertise computational thinking also strongly invoke employability, entry to the high-tech world, as a core incentive. Wing, herself, is not so overtly vocationalist in her talk.

There have been numerous meetings and workshops on computational thinking. One was organized by the National Research Council (NRC, 2010)—the research arm of Congress—which is sufficient to show the prominence and prestige accorded to the idea. Arguably the most important consequence of its rise to public awareness, computational thinking has become an explicit, endorsed idea behind substantial funding, on the order of multiple billions of dollars, over a hundred million dollars in the National Science Foundation alone (White House, 2016). The fact sheet for the initiative prominently features "computational thinking skills." Internationally, for example, the Royal Society of the United Kingdom took note, publishing an article by Wing (2008) in one of its journals and commissioning work on teaching computer science in school (Royal Society, 2010). Since then, the government of the U.K. has instituted a well-funded and clearly focused program to explore and actually create a good niche for computer science in pre-college instruction.

I want to gain some perspectives on Wing's work from the considerations I developed earlier. This will address two main issues having to do with the cultural properties of computational thinking as a movement, and with some of the intellectual heritage it has.

Cultural insularity

Wing is a computer scientist, and this shows pervasively in her writing. I hasten to add that every researcher has a home territory, particular views and experiences, cultural assumptions, and deep knowledge from their own community. That is not the issue. Instead, the issue is that the professional pursuit of understanding or creating a literacy—or anything that has similarly broad aspirations—cannot belong in any substantial degree to one of the standard professional disciplines: not computer science, not mathematics, not physics, not cognitive science, and not even education. The very point of a literacy—or of Wing's computational thinking—is, as she said, that it deeply impacts many if not all disciplines. It also requires all of the cognitive, educational, cultural, and other points of view that we can muster to query the nature and development of broad cultural and intellectual changes. My first principle, "literacy is a massive social/intellectual accomplishment," roughly marks the meta-terrain that it behooves us to develop and apply in order to make sound judgments about possible literacies and to help advance them optimally.

I will spend a little more time documenting Wing's computer-science-centric orientation before elevating for some perspective.

In her earliest paper (2006) and in later ones (e.g., 2014), Wing states and restates a core commitment to computer science per se in her pursuit. She says (2006) "Computational thinking is a grand vision to guide computer science educators, researchers, and practitioners as we act to change society's image of the field." This is a worthy pursuit for all great disciplines, but it is not a license for grand-scale, multi- or trans-disciplinary aspirations. A bit later, she says, "Rather than bemoan the decline of interest in computer science or the decline in funding for research in computer science, we should look to inspire the public's interest in the intellectual adventure of the field. We'll thus spread the joy, awe, and power of computer science, aiming to make computational thinking commonplace." In her 2014 piece, she again acknowledges this local-cultural aim. These are excellent points to recruit computer scientists to the agenda, but not so pertinent for the rest of us.

At middle levels, she mentions some core issues of computational thinking. She says, "Most fundamentally [computational thinking] addresses the question: What is computable?" Is this really so fundamental? Or is it only something core to computer science that others might benefit from knowing, but not core to a widespread intellectual agenda? How do we answer such questions?

Looking back at her technical definition: "Computational thinking is the thought processes involved in formulating a problem and expressing its solution(s) in such a way that a computer—human or machine—can effectively carry out." I see here a

slightly generalized version of "programming," which she makes clearer by talking more about how one might write down "the solution." The problem is that this model does not extend to some of the core tasks that she implicates for "computational thinking." Her definition is not what (non-computer) scientists do very much, except when they are writing computer programs. One essentially never figures out how to solve an important problem of a discipline and maps out exactly how it can be solved before actually doing the solving. Rehearse for yourself how any great historical scientist accomplished their feats, or how any example of modern basic research is working. For students, her most general image of computational thinking is something like "problem solving" (see documentation and commentary later). But her definition does not address how one solves problems in general, but only a very specialized class. "Programming" is core to my own computational literacy, but it is hardly sufficient to capture the broader idea—or else I would call it "programming" rather than "literacy."

I find throwaway examples of "cultural imperialism" equally charming and disturbing. Wing says "Computational thinking will have become ingrained in everyone's lives when ... [terms like] garbage collection take on the meaning used by computer scientists; and when trees are drawn upside down." "Garbage collection" is a wonderful, if esoteric (my judgment), technical issue in the implementation of particular classes of computer languages. How does it become an idea everyone should know about? Similarly, why should we all have to adopt culturally specific conventions, trees drawn upside down, to inscribe key ideas? Wouldn't any convention be fine?[15] For example, might not the hierarchy of boxes inside boxes be more practical for complex hierarchies, and more adapted to a broader range of uses? (I outlined and hierarchically organized this paper in Boxer; for such a task, trees, as computer scientists draw them, are impractical.)

The fundamental problem is not that any of these things to learn are necessarily bad things, but that Wing does not give us any way to think about them in the larger context. There are no *filters* for what from computer science should become "common knowledge." Why should garbage collection and one specific notational convention become part of computational thinking? Wing doesn't provide principles for *lift*, either. How does one abstract the many things she mentions from the computer science context to make them plausibly general? Finally, she does not elaborate *embedding*, how one places abstracted elements of computational thinking in the "destination" disciplines so as to make them important to mathematicians, physicists, or engineers? There are many assertions, but no principles for filtering, lifting or re-embedding. Such a serious enterprise as "computational thinking for all" deserves better.

The pool of references and allusions in all of her papers is completely dominated by computer science. The main intent of the paper by Grover and Pea, cited earlier,

[15] Our sixth grade motion students were quite clear on this: Conventions are convenient, but also arbitrary. See, diSessa (2004, p. 302).

was to begin filling in what is left out. To take a case of special interest to me, Wing cites no literature from the well-developed and continuing line of work in which I sit, started by Papert, aiming to bring computational ideas, indeed, programming, to the wider population for general intellectual purposes. Almost fifty years of work relevant to her core enterprise, which includes scores of books and hundreds of research papers, deserves some consideration. The community well beyond computer science knows something about what she imagines we all will want to do.

The next section elaborates a substantial, and, to me, problematic gap in the perspective Wing and others in the movement bring to computational thinking.

A brief history of *higher order thinking skills* and their relevance to computational thinking

Reading Wing's papers, those of us who have been around for some time will get an acute sense of déjà vu. Wing's advocacy for computational thinking echoes far back in time, and through a huge amount of relevant literature. Mathematics educators will be especially sensitive to at least some part of this story. Indeed, the first thing I noticed was that Wing sounded like George Pólya, with regard to problem solving in mathematics and mathematics education. Here are some of the ideas from Pólya (1945) of which there are versions in Wing's writings:

Decomposition and recomposition: Such general processes are very salient in computer programming, but Pólya elaborated essentially the same idea in 1945.

Draw a figure: Wing's version appears more general—select/design an appropriate representation—but the core idea is in Pólya. He also talked about the importance of notations, which is muted in Wing's writings. I venture to say that the prototype for "an appropriate representation" for Wing is, tacitly, a representation in a relevant programming language. From the experience I've had in studying how people generate representational systems (e.g., diSessa, 2004), I don't believe the skills to do this can be well developed only in the context of computational representations.

Generalize: Pólya advises us that a more general problem may be easier to solve. This is next door to, if it is not the same as, Wing's "abstraction," which she emphasizes repeatedly as at the very center of computational thinking. In Wing (2014), after defining computational thinking, she proceeds directly to a section labeled "Abstraction is key." The final paragraph of that paper asserts the "transfer" of computational thinking to "any domain," using abstraction as the central example. "The educational benefits of being able to think computationally—starting with the use of abstractions—enhance and reinforce intellectual skills, and thus *can be transferred to any domain*" (emphasis added).

Plans: Pólya emphasizes the power of designing a plan and carrying it out effectively. Wing's very definition of computational thinking is being able to design and record a plan (program) that solves a problem. But, as I pointed out, it is

questionable whether that really is the essence of much good and creative scientific thinking.

Even what might be called affective components of problem solving are anticipated in Pólya. Wing says, "Computational methods and models give us the courage to solve problems and design systems…." Pólya talks about determination, hope, and success.

It would be a good exercise to play this game more carefully and extensively, but here is the main point. In 1945 Pólya anticipated a lot of what Wing is putting forward, and more. In what ways is Wing's list better, more effective, more complete, and better justified than Pólya's? His work had decades of follow-up in educational work. Why is that work not cited, or recruited (or even denied!) in the task of finding the powerful general ideas that computational thinking can convey? Why aren't Wing and her followers drawing on and improving this and other relevant lines of work?

Let me frame the issue in a more general way. Wing's strong rhetoric about the general power of computational thinking draws very heavily on what I might call "the siren call of *higher order thinking skills*,"[16] HOTS, for short. This is an ancient and amazingly persistent meme, dating back at least to Plato, who, incidentally put mathematics as the home discipline for HOTS. He said, "Those who are by nature good at calculation are, as one might say, naturally sharp in every other study, and ... those who are slow at it, if they are educated and exercised in this study, nevertheless improve and become sharper than they were." (Grube, 1974, p. 178)

Jumping ahead twenty plus centuries, arguably the first highly visible and modern form of HOTS arrived in the early part of the 20th century. Stanic, 1986, on whom I draw here, provides a useful précis of the history of *mental discipline* and related ideas with respect to mathematics education. "Mental discipline" was a common moniker, but the general competence was often decomposed into supposedly general "faculties," which were, in turn, best developed in certain disciplines. Mathematics, of course, featured prominently as a host discipline. But so did other classical disciplines, including learning Latin and Greek. Now, it seems, Wing has put computer science as the prime locus for learning such general skills. Abstraction, Wing's candidate for a central "faculty" does not appear in early lists of faculties, as far as I could find. But it bears a strong family resemblance to things that were listed, and the issues concerning its legitimacy and power are the same.

According to conventional lore, Thorndike dealt a crushing blow to "mental discipline," and "faculty psychology" with a massive empirical study in the mid-1920s. Thorndike (1924) concluded, "… the intellectual values of studies [read

[16] The terminology might be a little abused in this context. HOTS might be reserved for the social movement that is explicitly affiliated with this banner. But, I need a more general framing, and HOTS seems to do that work reasonably.

"disciplines"] should be determined largely by the special information habits, interests, attitudes, and ideals which they demonstrably produce. The expectation of any large differences in general improvement of the mind from one study [discipline] rather than another seems doomed to disappointment." (p. 98) In other words, disciplines are powerful, but one should not expect them—most certainly not any *one* of them—to have substantial, general consequences.

Earlier, Thorndike and Woodworth (1901) made a comment that I think was especially insightful concerning faculties. They said, "[I]t is misleading to speak of sense discrimination, attention, memory, observation, accuracy, quickness, etc." This is so because, "[M]ultitudinous separate individual functions are referred to by any one of these words. These functions may have little in common" (p. 249). This resonates strongly with one of the prime results of my own studies of conceptual change. One of the main reasons that learning new concepts is so difficult is that they are, in fact, "multitudinous." Concepts are not monolithic, and a core difficulty is generating the separate forms and coordinating them properly.

Let us focus on Wing's core "faculty," abstraction. In the first place, it is a bit hard to know how one can pry this idea out of the possession of mathematics, the queen of abstract sciences, or why one should do that. But, Thorndike's observation goes deeper. Consider a brief thought experiment: Is it reasonable to consider abstraction to be the same thing in mathematics, physics, and computer science? I think not. Mathematical abstraction (let's call it, inferential abstraction) occurs in order to build conceptual worlds where a small set of attributes fully define entities, resulting in a substantial inferential fabric—a family of basic ideas and secure inferences (proofs) from them to other ideas (theorems). Abstraction in physics (empirical abstraction) is taking a look at the world and finding in it new things that cut away certain details, but build on others that might initially be completely ignored, in order to create core models that apply across a very wide range of circumstances. Mathematicians do not, in general, need or use the skill of "peeling away" from the world as it exists, nor digging through the difficulties of finding out how the world is in the first place, nor do they have the constraint of confirming empirically that the world admits in a certain abstraction, usually within prescribed limits.[17] Abstraction in computer science (practical abstraction) resides substantially in peeling away the irrelevant particulars of an implementation so that one only need think about the features of a piece of it that are essential for a given use—its "contract" with the rest of the program. In what respects can we guarantee that all these abstractions are similar enough that any one (say, abstraction in computer science) strongly facilitates abstraction in other fields?

[17] Many, even most, excellent mathematicians I know admit to not understanding physics. I think they are right (and I happily admit to the converse). But, it is difficult to understand this if the core process, "abstraction," is at the core of both disciplines.

Provocatively, a version of the idea of abstraction (perhaps "decontextualization" is a better descriptor of this version) appeared in the theory of literacy as a central power and differentiator compared to the intellectual capacities of pre-literate societies (Goody, 1977). The idea is that writing allows thinkers to separate themselves from the intricate details of lived context, which separation promotes a whole new and abstract level of thinking. However, more recent views of literacy have either ignored or argued against the idea of such central cognitive powers for literacy (Street, 1995; see Scribner & Cole, 1981, for a cautionary tale on how narrow versions of literacy—"reading and writing"—may not convey the kind of power that intellectual practices in which these skills are embedded may well convey).

After Pólya, the next major impetus for HOTS was arguably Newell and Simon's (1972) monumental accomplishment, *Human Problem Solving*. Following on that was a huge blossoming of study of problem solving, mainly as it regards individual disciplines (mathematics being a prime example), although there was a substantial accompanying, but usually less rigorous, blossoming of curricular and commercial work purporting to teach domain-general problem solving skills. Mathematical problem solving was studied from multiple perspectives and by a powerful group of educational researchers. It is problematic to summarize such complex intellectual histories in a couple of sentences, but that is all I can do here. I believe that the net result of the movement was that learning mathematics was not radically transformed. Problem solving does not seem to be critically powerful, even in a single discipline let alone transformative across disciplines.

Of course, problem solving has not disappeared, just as the "devastating blow" rendered by Thorndike's empirical work did not actually stamp out the ideas of mental discipline or faculty psychology. But the big picture of learning mathematics did not change. What remains is that the single most powerful element in gaining mathematical power is to understand deeply the concepts that the field deploys.

Not incidentally, Alan Schoenfeld, one of the prime movers of the mathematical problem solving movement, identifies the same factor that Thorndike identified, multiplicity, as a primary impediment to learning problem solving. All of the general heuristics and principles need specific consideration and learning in order to make them work across a wide range of circumstances (Schoenfeld, 1985).[18]

To reprise and briefly elaborate, I see Wing as skating on the psychologically deep, ancient, and amazingly persistent meme that supports substantial power for problem solving and other "domain general skills," HOTS, as I've rendered the

[18] Schoenfeld is still optimistic about the future of problem solving in mathematics education. I also have my own version of higher order competencies (knowledge about knowledge) about which I am optimistic. However, the big picture assumption of a revolution in problem solving as a general capacity has been seriously hobbled.

larger frame here. But the history of HOTS is complex, spotty, knotty, and devoid of consensus on what they are, how "teachable" they are, and whether they are effective. Is it responsible for a leader of a movement built on the siren call of HOTS *not* to mention the issues with it, and *not* to urge us to help resolve them? Wing has consistently reported advances and problems for computational thinking as if the HOTS-relevant intellectual history did not exist. Instead, she says that we need to figure out how to sequence programming learning and make it more broadly approachable and engaging; "we," apparently, do not need to demonstrate its general power, against the tide of history. It is easy—and perhaps appropriate—to acknowledge that a leader of a social movement may simplify and ignore complications. But one would still hope that the larger movement could address the issues. I see few signs of this. I just read an article about teaching of computer science and "computational thinking" at my home institution. Abstraction and its general power was a prime example, and the idea was used without the slightest self-consciousness. If there is good and deep research going on behind core ideas in the computational thinking movement, it is difficult for me to find,[19] and it is also virtually invisible in the public presence of the movement.

Coding

There is an undeniable boom in public interest in "coding,"[20] and a concomitant boom in commercial and non-commercial instruction; coding is both big business and a blossoming grassroots, altruistically oriented enterprise.

A serious memetic analysis of this movement would be fascinating. I can offer only tidbits here.

> **Meme 1:** Equity and empowerment are highly visible, particularly on the more grassroots side of things. "Black Girls Code" (and many other similarly directed organizations); advertisements of targeting in, and success in, minority and impoverished areas of the U.S. and the world.

I had a personal experience of the depth of this meme recently when a reporter came to talk to me about coding academies. She was evidently unhappy with my lukewarm assessment of the movement. When I probed for the reason for her own enthusiasm for coding, it came out that she had recently enrolled in an academy and felt personally uplifted by the experience, as if a prohibition had been lifted on

---

[19] The NRC (2010) report on computational thinking addresses some gaps, including the lack of attention to prior, concurrent, extensive and very similar movements, such as the Papert/MIT line of work. However, (1) that report, of course, does not do the actual work it proposes to be important, (2) later papers, such as Grover and Pea (2013), cited earlier, suggest that much of that work is undone and forms a continuing, important agenda that is broadly unacknowledged, and (3) the report did not substantially bring the history of relevant scholarship critically to bear on the public face of computational thinking.

[20] It is interesting how terminology shifts—for example, "programming" becomes "coding"—when the cultural resonance and contextualization shifts.

her, and a wonderful intellectual world had opened up. It seemed the source of liberation was two-fold. As a writer with a liberal arts background, all of a sudden, she had access in an apparently successful way to modern technoscience. In addition, she felt the movement was opening up pathways for women that didn't exist in the past: equity and liberation.

> **Meme 2**: Computational thinking – Although not pervasive in my sampling, a lot of coding instruction is accompanied by computational thinking rhetoric, sometimes with phrases that might have been lifted from Wing's writings. I suspect that computational thinking is not just an advertisement of "competitive advantage" of some academies, but one of the backdrop memes driving the movement.
>
> **Meme 3**: Arguably the most visible meme one sees in the coding movement is *vocationalism*: "jumpstart your career"; "your teen's future in coding;" "the thrill of a start-up culture;" advertising the most current and well-known professional languages.

The attractiveness of the vocational side of coding is understandable. Few things are as important and salient in modern life as finding a good, well-paying job. However, I think it is well worth being skeptical of memes that are so self-evident in contemporary society that few would bother to data-check their real status.

While I make no pretense to expertise or definitive conclusions, I did bother to go to the U.S. Department of Labor's website to see what I could find. Here are some observations (unless otherwise stated, statistics are from the Bureau of Labor Statistics website and dated 2015):

All jobs classified under "computers and IT" constitute only about 2.4% of the labor force.

Then I looked more particularly at the three most programming-related subcategories of computers and IT. Programming, per se, is not a very positive-looking job category. It comprises 0.2% of the U.S. labor force, and is projected to *decline* 8% over the next decade, owing mainly to foreign competition.

Software developers and web developers provide a more positive picture. Both are growing much faster than average. But both, together, are significantly less than 1% of the labor force and will not even reach 1% over the next decade, under the Department's projections of growth.

For comparison, I looked at the healthcare sector. It turns out nursing alone, in its various subspecialties, constitutes a proportion of the labor market that is comparable to all computing and IT categories combined. But the subcategory of registered nurses by itself is much larger than the three "programming" categories, mentioned above, combined. Some subspecialties in the nursing category are

growing faster than any of the three "programming" categories. Interestingly, the fastest growing job category listed by the Bureau of Labor Statistics happens to be "wind turbine technician" (although the category, at present, is tiny), which is suggestive of some important trends. See also, "The U.S. wind industry now employs more than 100,000 people" (2017).

This all suggests that the vocational value of programming, per se, might be overblown and more symbolic than functional. I believe this is true, but there is a deeper issue, which can be seen from the top level of my computational literacy analytic. I call it "the mire of vocationalism."

Literacies, by nature, are not narrow channels. They work always (I believe) by recruiting a broad range of intellectual enterprises to a common expressive system. One doesn't learn reading and writing in school based on a narrow, vocationalist targeting of skills. Textual literacy is not just for professional writers. And you have to acquire many important and specific skills, beyond the basic literacy skills you learn in school, to become a poet or journalist. Furthermore, we don't teach reading and writing only within a particular vocational track. Reading and writing specialized for physicists, or for mathematicians, or for nurses, or for engineers? That does not work. It will not work for computational literacy.

There is a broader version of vocationalism, not learning programming to become a programmer, that holds more promise. By now, there is a decently well-developed literature that deals with how computation and mathematics are really integrated into work situations. The upshot is that calculation and its computational counterpart, programming, are surprisingly seldom seen in work situations. Instead, what is important is understanding the higher levels of what calculation or computation can accomplish, their limitations, their in-built assumptions about the world, signs of failure, and what might be done to contextualize algorithms better, or even change them to suit local needs. I am not sure any one word captures these perspectives, but mathematical or computational "modeling" is a decent start. Noss (1998) offers a brief and easy-to-understand introduction. In any event, these directions define part of a broader and more legitimate vocationalism, which is simply not "coding," per se.

One of my greatest disappointments in the formative days of computers and education was the selection of Pascal as the relevant language for the AP test, signaling the cultural dominance of computer science and computer scientists in the definition of what computing should be. Pascal was a learning language developed by and for the computer science community, and the AP test, thereafter, tracked evolving, modern, professional languages: C++, Java, Java 5. The early choice of Pascal as the de facto standard for programming horrified and shook me because it made me realize what shape programming would take in our culture, at least for a long time. It would be narrowly vocationalist and techno-centric.

Two final points: Most coding academies and related entities are not really about anything other than programming, per se, although some might claim to include "computational thinking." They don't concern, for example, the mathematics of motion or any equivalent set of ideas. This seems to me akin to learning a language by learning its grammar and never talking about anything important using it. What is most engaging about textual literacy is reading a good book, or writing about something you care about. Our sixth grade students were very clear on this: Programming was engaging precisely in the wonderful things they did with it. So, how do you make programming and computer science engaging? Do something interesting and important with it.

Overall, coding academies may be serving some aims related to computational literacies, possibly "ticking off" formal or informal prerequisites for more interesting and important work. Or, for some, serving as an interest on-ramp to technoscience (which appeared to be the case with the reporter that visited me about coding academies). But they are most definitely not about transforming intellectual domains (e.g., motion), nor inducting students into civilization-wide "great literature," nor freeing personal creativity and expression, nor even about "modeling," as discussed just above.

My last point is that I have found the computer environments and instructional strategies used by many coding academies to be archaic and, again, as far as I can see, completely without cognizance of the things we learned about teaching programming in the Logo community and elsewhere. It is strongly ironic that current learning environments are mostly much worse than what we had almost fifty years ago, now that, finally, there is a mass movement toward learning computation.

Some of the reasons for this retrograde move stem from narrow vocationalism, for example, using professional programming environments that can have devastating consequences on learning. I have seen those consequences in depressing detail, owing to the work of students that have been involved with coding academies. Another reason is likely historical ignorance and a general faith that "we are doing something essentially new—there's no point consulting history."

Synthetic Review and Some Practical Advice

The perceived promise of computation to revolutionize mathematics (and, more generally, STEM) education has a near half-century history. Computers and computation have inspired researchers' and educators' hope that these instruments can make a special contribution toward achieving long-delayed goals. Some such goals are practical: to teach more and different things (e.g., "more modern" topics), and more efficiently and faster. Some goals are more inspirational: to have mathematics come to feel more natural, relevant, and less intimidating. Other goals are moral or ethical: to be more inclusive.

This paper dips into the set of "big pictures" that can organize work toward realizing goals such as those above. I introduced one possible benchmark, computational literacy, and a set of principles concerning it. The principles help define the benchmark, but they also serve two other important functions. They serve as an analytical framework to compare and contrast with competitive big pictures. As I mentioned, computational literacy seems poorly differentiated in the public eye from a more recent and extremely prominent candidate, computational thinking. In addition, these principles mark scientific loci of inquiry that can validate and enhance progress toward achieving the goals of various big pictures, or, in complementary manner, undermine their credibility, while simultaneously suggesting potential improvements.

Review of Principles

I noted first that **literacy-scaled accomplishments are massive social and cultural accomplishments**. This suggests both that we should set our sights high, and that we need also to be modest about what any one of us, or any movement, can accomplish. Present times, present curricula and sensibilities about instruction and learning do not set a definitive frame. While it is laudable to try to be effective sooner rather than later, we need to balance these considerations with anticipation of very substantial future profit from changing current assumptions and sensibilities. In particular, if there are no recognized landmarks of substantially changed teaching and learning, few will be inspired to step outside the bounds of contemporary practice. I am in favor of being bold, trying out things with great promise despite a potentially uncomfortable fit with the present. The proposal to teach vector calculus almost got our proposal concerning the mathematics of motion to sixth graders rejected. But, in the end, it was one of our easiest and most satisfying accomplishments. At the same time, an "accidental" cultural trend enhanced some aspects of our work. Entirely independent of the motivations we had for working on the mathematics of motion—independent of the anticipation of cognitive simplicity that would be displayed and independent of the larger frame of computational literacy—curricular standards across the country began incorporating motion as an elementary and middle school topic. Some of our work collaborating with other groups was strongly reinforced by teachers' felt needs when they were suddenly charged with instructing motion, but with no preparation or appropriate curricula to address that demand. So one can also take advantage of present-day memes and movements to take steps toward a larger and possibly independent agenda.

Social change, specifically relevant to the emergence of a new literacy is, I believe, the least scientifically understood locus among those referenced in my principles. So, I felt it necessary for rhetorical purposes to augment the top level of this principle with a "toy" model of social change, identifying elements of culture that are relevant to the emergence of a computational literacy, or similar large-scaled social change. I talked mainly about **memes** and **movements**, but also introduced **values, sensibilities**, and **sensitivities** (collectively MMVSSs). A key lesson is that

contemporary MMVSSs may help propel us roughly in the direction we want to go, but surreptitiously introduce distractions (e.g., a narrow vocationalism) and ignore critical dimensions of change (e.g., change in the fundamentals of what is taught, when).

Next, there are the "four Rs," which add detail and specific foci for a literacy-relevant agenda.

Re-mediation – The representational infrastructure for our civilization is changing dramatically. Dynamic and interactive representations on computers, along with the ability to design and enact specialized representations on demand and often quickly means that intellectual changes easily on the scale of what algebra or calculus brought us are almost certainly in the offing. I have taken a particular line on this by aiming mainly to develop and build on generic resources (a generic computational medium, including high-end compositional resources—programming—such as Boxer provides). But other paths exist, for example exemplified in the vibrant community that built up around dynamic geometry systems (Sketchpad, Cabri Geometry), and similar families of specialized representations for individual topics (such as Tabletop for statistics), or, more in the direction of Boxer, things like Mathematica or Python notebooks.

Reformulation – I introduced the possibility of very substantial changes in what, when, and how we teach subject matter based on an improved understanding of native human intellectual resources: reformulation. For those who know my early work on turtle geometry with Hal Abelson (Abelson & diSessa, 1981), much of that stands as productive reformulation even independent of computational re-mediation. The cognitive foundation for turtle geometry is the use of body-based dynamic representation, or the body as a modeling language, as a much better basis for learning about geometry than that used by conventional static and proof-centered approaches. Papert called it a "body syntonic" approach. A clean and impressive demonstration of reformulation without re-mediation is teaching turtle differential geometry to early and middle elementary school students without computer mediation (Lehrer, Holmes, Taimina, & Henderson, 2016). But motion as approached in our sixth grade course is different. This reformulation (e.g., motion construed in discrete form) without computational support is minimal and wasteful. It is an easy step to add computation, and the resulting combination is dramatically superior along multiple dimensions to the one without computation.

I deliberately backgrounded scientific progress on "cognitive simplicity" as too complicated and distinct to join here with the agenda of this paper. However, it has strongly influenced my research group's work. For example, diSessa (2017) documents a stunning example of student learning when based on the instructional possibilities of virtually unknown intuitive knowledge (unknown, that is, in the broader educational and cognitive research community). While these intellectual pursuits are not currently very prominent in educational research, I recommend them to those who wish to pursue anything like computational literacy.

Reorganizing the intellectual terrain – This principle creates a landmark for deep change in teaching and learning, concomitant to the more general focus on big cultural changes. Teaching mathematics will and should look very different, given *any* literacy-scaled change. If you are stuck trying to optimize current instruction, say, in linear algebra, at the same age levels, only "using computers," you are not playing the game I recommend. Of course, as I remarked earlier, the "self-evident" epistemology of the times will get in the way, as well as, on occasion, providing "hooks" and cultural resonances for advance. I have repeatedly emphasized that this is a difficult agenda. However, orienting toward big, long time-scale rewards is what we need to do.

The slogan "**a literacy needs a literature**" fits just here. A reorganized intellectual terrain will show up in new landmark documents (textbooks, or newer and very different interactive forms), or in a revised list of civilization's "great ideas." The slogan and its larger-scaled recognition of major content-specific shifts in how we think about particular subject matter, and, by extension, about curriculum, also warn against engaging computation without respecting and engaging the great intellectual accomplishments of our civilization *beyond* that directly related to computers.

Revitalizing the ecology of learning activities – I am deeply troubled by the fact that much of our mathematical and scientific instruction is organized around small, impersonal, and denatured problems. I am equally encouraged and motivated by what happened in our motion class. Programming computer games is not a sufficiently broad prototype (it doesn't describe many of the successful activities in our motion course), but adding design and authentic research (where "authentic means both "true to the discipline" and also "fitting students' interests and current conceptual landscape") to the list does go a long way.

Here is one place where engagement, interest, and, by necessary extension, equity—respecting important diversity in classrooms—is strongly visible in the program. Some theoretical work concerning engagement in our motion class appears in Azevedo, diSessa, and Sherin (2012). Here is also where the equivalent of "writing as well as reading" in a computational literacy ("writing" = programming) pays strong dividends. Research that look superficially like computational literacy (new representational infrastructure, perhaps even steps toward significant reformulation) might diverge just at this point.[21]

---

[21] A good study of a program whose rhetoric looks very similar to the way that I describe our mathematics of motion class—for example, changing representational infrastructure, reformulation (a discrete epistemology) and consequent democratizing access to calculus—is the SimCalc Project (Kaput, 1994; Stroup, 2002). That project also undertook to bring the study of motion into the computer age. However, in terms of deeply engaging student interests and creativity in personally motivated activities, SimCalc's tenor is quite distinct. SimCalc also

Applying the Principles

Using these principles to provide helpful comparative analysis of competing perspectives was illustrated here by considering two contemporary trends in computer-oriented instruction. The first may well be the most important movement in a decade or two toward big changes in the use of computers in learning: computational thinking. It swamps the social visibility of computational literacy, and its connection to contemporary MMVSSs is more vivid.

On the other hand, invoking literacy principles puts a critical light on computational thinking. Perhaps most central is the lack of orientation toward domain-specific adaptation. At least three of my principles concern this. First, **re-mediation** underscores the general fact that any representational system is better adapted for some things and less so for others. This needs to be explored on a domain-by-domain basis, and work must be done to adapt what is done with computational representations to each domain. Our success in teaching vectors came substantially from how they were fit into the medium, even if we benefitted from more generic properties of computational media.

Similarly, **reformulation** is also a domain-by-domain issue. One needs to find the productive roots of different ways of thinking about each domain. Certainly there are some generalities, but so far as I have seen (and a lot of literature agrees), one needs to understand how learners construe particular domains, and how they *may* profitably construe them differently.

In contrast, computational thinking emphasizes a fairly extreme version of domain generality. There is no talk whatsoever about **reorganizing domains** or **new literatures** coming from computational thinking advocates. More, rhetoric concerning computational thinking emphasizes a kind of generality ("educational benefits of being able to think computationally … enhance and reinforce intellectual skills, and thus can be transferred to any domain") that I believe is, at best, scientifically suspect, even if it propels the movement forward via MMVSS cultural resonance.

My analysis of the coding movement proceeded similarly. While the movement provides a solid socially widespread impetus that is critical for those of us concerned with computational literacies (although some of its powerful memes are suspect), it ignores—if it does not counter—most of the concerns that define computational literacy.

---

articulately chose to reject using computational media and programming in favor of specialized, narrowly targeted and non-student penetrable representations (see Roschelle, Kaput, Stroup, & Kahn, 1998). The near-future orientations of the SimCalc Project have served it very well. However, that leaves a larger agenda unaddressed.

This might be the place to consider how to optimally combine the energy and insights of computational thinking and coding movements with the insights and directions of computational literacy. But, given the length of this essay, that is a topic for the future.

"Practical" Advice

I close with some epigrammatic and action-oriented syntheses of arguments presented in this article along with a few more strategic suggestions.

Think big; orient toward the best that you can imagine. If you have your own big picture, seek to develop it empirically and theoretically, and to test it seriously against other big pictures of computers in mathematics learning. Being strategic and accountable to contemporary memes may be necessary and appropriate. But a clearer vision of where we might go, beyond our current state, can take us further.

The most serious impediment to anything on the scale of computational literacy is the blinding obviousness of unquestioned assumptions about the present, accepting that what is now will always be so.

Question ready epistemological assumptions about what is necessary to teach and what is beyond reach at particular ages, or what fits where in the global sequence of topics in mathematics education. Background "methods for teaching" compared to the framing and approach (e.g., cognitive simplicity, synergy with new representational forms) of the very mathematics, itself.

Get to know computational media as deeply as you can. My preference is learning programming in a suitable computational medium—or as close to that as you can get. But, in any case, look to gain acquaintance with the most epistemologically rich computer systems you can find. By "epistemologically rich," I mean having legitimate—but likely as-yet unrealized—consequences for the mathematics we can experience and might teach. I learned a tremendous amount by hanging out with computer scientists (several years in MIT's Artificial Intelligence Laboratory, and a decade in the Laboratory for Computer Science), but I wouldn't take them as authorities on what computation means for mathematics education. Cultivate your computational autonomy.

Combining the last two points, what proved extremely productive to me was to pursue both possible cognitive simplicities and synergistic re-mediations in order to approach, *for yourself*, mathematics that is vague for you. I taught myself some core parts of differential geometry, with which I had only a dim impression, in a few days using body syntonic approaches to the topic. I discovered some number theory that I had never learned and geometric theorems that I never knew existed by playing with simple turtle programs. This is a new world; don't assume it is just slightly different from—or maybe even just "an approach" to teaching—"the usual stuff." The relationships of computation to any of these specialties requires inquiry;

I emphasized the multifaceted nature of computational literature. There is no single recipe for how computation changes a field or subfield.

If your pursuits take you in different directions than I suggest here, that will enrich the horizon for all of us. If they parallel or extend what I and others who are focused on the big picture have already done, perhaps we can converge sooner than might be expected.

I read the pursuit of a protean big picture for computer use in mathematics education as a noble and largely selfless pursuit. None of us, nor any single subgroup, can change civilization by ourselves. But, we can be more-valuable-than-average and more than unwitting participants, bringing some order to a chaotic social process driven by unquestioned but questionable assumptions and lack of big-picture aspirations.

Acknowledgments

Thoughtful comments and suggestions, which led to many improvements, were provided by Bruce Sherin, George Gadanidis, Rich Lehrer, Celia Hoyles, Richard Noss, and Melinda diSessa. The decades of work that led to the perspective presented here could not have happened without the support of research grants from the National Science Foundation and the Spencer Foundation, or without dozens of students, co-workers and teachers who did most of the work.

References

Abelson, H., & diSessa, A. A. (1981). *Turtle Geometry: The computer as a medium for exploring mathematics*. Cambridge, MA: MIT Press. Subsequently produced in paperback (1985); editions translated into Spanish, Italian, Japanese, Hungarian and Polish.

Azevedo, F. S., diSessa, A. A., & Sherin, B. (2012) An evolving framework for describing student engagement in classroom activities. *Journal of Mathematical Behavior*, *31*, 270‑289. doi:10.1016/j.jmathb.2011.12.003

Bureau of Labor Statistics (2015). Occupational Outlook Handbook. www.bls.gov/ooh/

Dalton, B., Ingels, S.J., & Fritch, L. (2016). High School Longitudinal Study of 2009 (HSLS:09) 2013 Update and High School Transcript Study: A First Look at Fall 2009 Ninth-Graders in 2013 (NCES 2015-037rev). U.S. Department of Education. Washington, DC: National Center for Education Statistics. Retrieved [3/20/2017] from http://nces.ed.gov/pubsearch.

diSessa, A. A. (1995). Designing Newton's laws: Patterns of social and representational feedback in a learning task. In R.-J. Beun, M. Baker, & M. Reiner (Eds.), *Dialogue and interaction: Modeling interaction in intelligent tutoring systems* (pp. 105-133). Berlin, DE: Springer-Verlag.

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

diSessa, A. A. (2004a). Meta-representation: Native competence and targets for instruction. *Cognition and Instruction*, *22*(3), 293-331.

diSessa, A. A. (2004b). Principles of teaching physics with computers. In E. Redish & M. Vicentini (eds.), *Proceedings of the International School of Physics "Enrico Fermi": Research on physics education* (pp. 157-173). Amsterdam, NL: ISO Press/Italian Physics Society.

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. , Hammer, D., Sherin, B., & Kolpakowski, T. (1991). Inventing graphing: Meta-representational expertise in children. *Journal of Mathematical Behavior*, *10*(2), 117-160.

Fewer Black Students Are Taking Calculus in High School (2017). *U.S. News & World Report.* Retrieved [3/20/2017] from www.usnews.com/news/stem-solutions/articles/2015/07/06/fewer-black-students-are-taking-calculus-in-high-school.

Goody, J. (1977). *The domestication of the savage mind*. Cambridge, UK: Cambridge University Press.

Grover, S., & Pea, R. (2013). Computational thinking in K-12: A review of the state of the field. *Educational Researcher*, *42*(1), pp. 38–43. doi: 10.3102/0013189X12463051

Grube, G. M. A. (Trans.). (1974). *Plato's Republic*. Indianapolis, IN: Hackette.

Hmelo-Silver, C. E., & Duncan, R. G. (Eds.). (2009) Special Issue: Learning progressions. *Journal of Research in Science Teaching*, *46*(6), 605-737.

Kaput, J. (1994). Democratizing access to calculus. In A. Schoenfeld (Ed.), *Mathematical Thinking and Problem Solving* (pp. 77–156). Hillsdale, NJ: Lawrence Erlbaum Associates.

Klein, D. (2013). A brief history of American K-12 mathematics education in the 20th Century. In J. Royer (ed.) *Mathematical cognition* (pp. 175-225). Greenwich, CT: Information Age Publishing.

Lehrer, R., Holmes, A., Taimina, D., & Henderson, D. W. (2016, April). Defining on plane, cylinder, sphere and hyperbolic plane. Paper presented as part of a symposium *Studies of Children's Emerging Sense of Space and Measure.* The annual meeting of the National Council of Teachers of Mathematics Research Conference, San Francisco, CA.

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

Noss, R. (1998). New Numeracies for a technological culture. *For the Learning of Mathematics*, *18*(2) 2-12.

Pólya, G. (1945). *How to solve it*. Princeton, NJ: Princeton University Press.

Roschelle, J., Kaput, J., Stroup, W., & Kahn, T. M. (1998). Scaleable integration of educational software: Exploring the promise of component architectures (Special Issue on "Educational Authoring Tools and the Educational Object Economy"). *Journal of Interactive Media in Education*. 1998(*2*), part 6. doi: 10.533/1998-6.

Royal Society (2010). www.royalsociety.org/education-policy/projects/

Saxe, G. B. (2014). *Cultural development of mathematical ideas*: *Papua New Guinea studies*. NY, NY: Cambridge University Press.

Schoenfeld, A. H. (1985). *Mathematical problem solving*. Orlando, FL: Academic Press.

Scribner, S., & Cole, M. (1981). *The psychology of literacy*. Cambridge, MA: Harvard University Press.

Sherin, B. (2001). A comparison of programming languages and algebraic notation as expressive languages for physics. *International Journal of Computers for Mathematical Learning*, *6*, 1–61.

Stanic, G. A. (1986). Mental discipline theory and mathematics education. *For the Learning of Mathematics, 6*(1), 39-47.

Street, B. V. (1995). *Social literacies*. London, UK & NY, NY: Longman.

Stroup, W.M. (2002). Understanding qualitative calculus: A structural synthesis of learning research. *International Journal of Computers for Mathematical Learning*, *7*(2), 167–215.

The U.S. wind industry now employs more than 100,000 people (2017). *The Washington Post*. Retrieved [4/20/17] from www.washingtonpost.com/news/energy-environment/wp/2017/04/19/the-u-s-wind-industry-now-employs-more-than-100000-people/.

Thorndike, E. L. (1924). Mental discipline in high school studies. *Journal of Educational Psychology*, *1*(5) 1-22. doi: 10.1037/h0075386.

Thorndike E. L., & Woodworth, R. S. (1901). The influence of improvement in one mental function upon the efficiency of other functions (I). *Psychological Review*, *8*, 247-261.

White House (2016). FACT SHEET: President Obama announces computer science for all initiative. Retrieved [4/20/17] from https://obamawhitehouse.archives.gov/the-press-office/2016/01/30/fact-sheet-president-obama-announces-computer-science-all-initiative.

Wilensky, U., & Papert, S. (2010). Restructurations: Reformulations of knowledge disciplines through new representational forms. In J. Clayson & I. Kallas (Eds.), *Proceedings of the constructionism 2010 conference* (pp. 97-111). Paris, FR, Aug 10-14.

Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, *49*(3), 33-35. doi: 10.1145/1118178.1118215

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

Wing, J. M. (2011). Research Notebook: Computational thinking—What and why? *The Link*. Retrieved from https://www.cs.cmu.edu/link/research-notebook-computational-thinking-what-and-why

Wing, J. M. (2014). Computational Thinking Benefits Society. *40th Anniversary Blog of Social Issues in Computing*. Retrieved from http://socialissues.cs.toronto.edu/index.html%3Fp=279.html