CrossMark

Defining Computational Thinking for Mathematics and Science Classrooms

David Weintrop[1,2] · Elham Beheshti[3] · Michael Horn[1,2,3] · Kai Orton[1,2] ·
Kemi Jona[2,3] · Laura Trouille[5,6] · Uri Wilensky[1,2,3,4]

Abstract Science and mathematics are becoming computational endeavors. This fact is reflected in the recently released Next Generation Science Standards and the decision to include "computational thinking" as a core scientific practice. With this addition, and the increased presence of computation in mathematics and scientific contexts, a new urgency has come to the challenge of defining computational thinking and providing a theoretical grounding for what form it should take in school science and mathematics classrooms. This paper presents a response to this challenge by proposing a definition of computational thinking for mathematics and science in the form of a taxonomy consisting of four main categories: data practices, modeling and simulation practices, computational problem solving practices, and systems thinking practices. In formulating this taxonomy, we draw on the existing computational thinking literature, interviews with mathematicians and scientists, and exemplary computational thinking instructional materials. This work was undertaken as part of a larger effort to infuse computational thinking into high school science and mathematics curricular materials. In this paper, we argue for the approach of embedding computational thinking in mathematics and science contexts, present the taxonomy, and discuss how we envision the taxonomy being used to bring current educational efforts in line with the increasingly computational nature of modern science and mathematics.

Keywords Computational thinking · High school mathematics and science education · STEM education · Scientific practices · Systems thinking · Modeling and simulation · Computational problem solving

Introduction

> By 2020, one of every two jobs in the "STEM" fields will be in computing
> (ACM pathways report 2013)

The release of the Next Generation Science Standards (NGSS) places a new emphasis on authentic investigation in the classroom, including eight distinct scientific practices (NGSS Lead States 2013). While some of these practices are familiar to veteran teachers, such as "asking questions and defining problems," others are less well understood. In particular, the practice of "using mathematics and computational thinking" reflects the growing importance of computation and digital technologies across the scientific disciplines. Similar educational outcomes can be found in mathematics standards, such as the Common Core guidelines, which state that students should be able "to use technological tools to explore and deepen their

✉ David Weintrop
dweintrop@u.northwestern.edu

1 Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL 60208, USA

2 Learning Sciences, Northwestern University, Evanston, IL 60208, USA

3 Computer Science, Northwestern University, Evanston, IL 60208, USA

4 Northwestern Institute on Complex Systems, Evanston, IL 60208, USA

5 The Adler Planetarium, Chicago, IL 60605, USA

6 Center for Interdisciplinary Exploration and Research in Astrophysics (CIERA), Northwestern University, Evanston, IL 60208, USA

⚙ Springer

understanding of concepts" (National Governors Association 2010, p. 7). However, the inclusion of these practices, in and of itself, offers little guidance for teachers who will be required to realize them in their classrooms. Much of the difficulty stems from the fact that the practices collected under the umbrella term "computational thinking" (National Research Council [NRC] 2010; Wing 2006; Papert 1996) have not yet been clearly defined. This is especially true for their use in scientific or mathematical contexts as opposed to more general computer science settings. There is also active debate and discussion around key questions such as: How is computational thinking related to mathematical thinking, algorithmic thinking, or problem solving? How does it relate to the field of computer science? To what extent is computer programming involved? Does computational thinking always require a computer?

Our aim in this paper is to develop a more nuanced understanding of computational thinking specifically as it applies to the mathematic and scientific disciplines and the needs of high school teachers who are expected to prepare students for potential careers in these fields. Unlike most of the discussion on computational thinking to date, which emphasizes topics from computer science such as abstraction and algorithms, our approach to defining computational thinking takes the form of a taxonomy of practices focusing on the application of computational thinking to mathematics and science. This approach employs mathematics and science as meaningful contexts in which to situate the concepts and practices of computational thinking and draws on the ways mathematicians and scientists are using computational thinking to advance their disciplines. This more restrictive context allows us to more clearly characterize what computational thinking is in mathematics and science.

The taxonomy consists of four main categories: data practices, modeling and simulation practices, computational problem solving practices, and systems thinking practices. We describe each of these practices and their constituent components, and we describe what it looks like to demonstrate mastery of each practice. The contribution of this work is to provide an actionable, classroom-ready definition of computational thinking that draws on existing computational thinking scholarship and incorporates concepts specifically focused on mathematics and science contexts. In doing so, we provide a framework and shared language that can be used to bring mathematics and science instruction more in line with their increasingly computational nature. Further, in grounding our conception of computational thinking in mathematics and science, we narrow the scope of computational thinking away from generalities, providing a sharper definition that is distinct from computer science, yet still grounded in authentic, meaningful computational practices that are essential for students to master.

Why Bring Computational Thinking to Mathematics and Science Classrooms?

A primary motivation for introducing computational thinking practices into science and mathematics classrooms is the rapidly changing nature of these disciplines as they are practiced in the professional world (Bailey and Borwein 2011; Foster 2006; Henderson et al. 2007). In the last 20 years, nearly every field related to science and mathematics has seen the growth of a computational counterpart. Examples include Bioinformatics, Computational Statistics, Chemometrics, and Neuroinformatics. This rise in importance of computation with respect to mathematics, science, and the broader Science, Technology, Engineering, and Mathematics (STEM) fields has been recognized both by those within the STEM education communities and computer science education organizations (ACM/IEEE-CS Joint Task Force on Computing Curricula 2013). Bringing computational tools and practices into mathematics and science classrooms gives learners a more realistic view of what these fields are, better prepares students for pursuing careers in these disciplines (Augustine 2005; Gardner 1983), and helps equip students to be more savvy STEM citizens in the future. As Foster (2006), director of the Computation Lab at the University of Chicago, states, "all scientists will be adept at applying existing computational techniques" (p. 419). Further, the varied and applied use of computational thinking by experts in the field provides a roadmap for what computational thinking instruction should include in the classroom.

From a pedagogical perspective, the thoughtful use of computational tools and skillsets can deepen learning of mathematics and science content (Guzdial 1994; Eisenberg 2002; National Research Council 2011a, b; Redish and Wilson 1993; Repenning et al. 2010; Sengupta et al. 2013; Sherin 2001; Wilensky 1995; Wilensky et al. 2014; Wilensky and Reisman 2006). The reverse is also true—namely that science and mathematics provide a meaningful context (and set of problems) within which computational thinking can be applied (Hambrusch et al. 2009; Jona et al. 2014; Lin et al. 2009; Wilensky et al. 2014). This differs markedly from teaching computational thinking as part of a standalone course in which the assignments that students are given tend to be divorced from real-world problems and applications. This sense of authenticity and real-world applicability is important in the effort to motivate diverse and meaningful participation in computational and scientific activities (Blikstein 2013; Chinn and Malhotra 2002; Confrey 1993; Margolis and Fisher 2003; Margolis 2008; Ryoo et al. 2013). This reciprocal relationship—using computation to enrich mathematics and science learning and using mathematics and science contexts to enrich

computational learning—is at the heart of our motivation to bring computational thinking and science and mathematics concepts together.

A final motivation for bringing computational thinking into mathematics and science classrooms is to reach the widest possible audience and address longstanding issues of the underrepresentation of women and minorities in computational fields (National Science Foundation 2013). Currently, only a fraction of high school students have the opportunity to take a computer science course due to a lack of qualified teachers, inadequate facilities, or constraints in class scheduling. Embedding computational thinking activities in mathematics and science coursework directly addresses the issue of students self-selecting into (or out of) computer science classes, which has been a challenge long plaguing the effort to reach underserved youth (Margolis and Fisher 2003; Margolis 2008). It also avoids practical issues of fitting new classes into overcrowded school schedules and finding teachers to teach them.

Intended Audiences

One contribution of the work we present here is an actionable set of guidelines that can be followed to bring computational thinking into mathematics and science classrooms quickly and effectively. In choosing to span mathematics and science broadly, this taxonomy defines a shared language that can be used across classrooms and departments to help students understand the crosscutting nature and broad applicability of computational thinking. As such, the taxonomy we present in this paper is intended for a diverse set of educational stakeholders including teachers, administrators, curriculum designers, assessment developers, and education researchers. For teachers, our taxonomy is meant to provide a concrete, clearly delineated set of practices to guide classroom implementation and curriculum development. We hope to help teachers understand how they are already using computational thinking practices in their classes and to support them in more fully developing those aspects of their lessons. Our goal is not to radically change the existing practices of experienced teachers; instead, we want this taxonomy to serve as a resource for augmenting existing pedagogy and curriculum with more sophisticated computational thinking practices. For administrators and policy makers, the taxonomy is meant to help shape expectations and set priorities for mathematics and science education, particularly when it comes to preparing students for the demands of the twenty-first century. Understanding the increasingly computational dimensions of these fields might also help administrators allocate professional development resources according to teacher needs.

For curriculum developers and other designers of learning experiences, our hope is that the taxonomy will serve as a resource to address "what" and "how" questions that accompany the creation of new educational materials. With the increased importance of accountability in our schools, the need for accurate, validated assessments is essential. By providing a clear definition of what computational thinking in scientific and mathematical contexts includes, this taxonomy can be used as a resource for assessment developers who are tasked with creating the items that will measure these practices.

Finally, for education researchers, we view this work as both a theoretical and practical contribution to our understanding of the nature of science and mathematics education in our increasingly technological age. While the practices of science and mathematics have greatly changed over the past 50 years due to advancing technology, classrooms have not kept pace. By delineating the space of computational thinking with respect to these fields, we hope to provide a resource for other educational researchers to use in their efforts to modernize science and mathematics learning to better prepare students for the computational future that awaits them.

Background

In this section, we briefly review three literatures that have informed the taxonomy we present in this paper. First, we review the literature on computational thinking, situating it historically, and illustrating relevant recent efforts to operationalize the concept. We then discuss research focusing on bringing computational thinking into the classroom, a goal that is central to our research agenda. Finally, we review the growing role of computation in mathematics and science fields.

Computational Thinking

> To reading, writing, and arithmetic, we should add computational thinking to every child's analytical ability
> (Wing 2006, p. 33)

The driving theme behind the recent interest in computational thinking is that knowledge and skills from the field of computer science have far reaching applications from which everyone can benefit: "[computational thinking] represents a universally applicable attitude and skill set everyone, not just computer scientists, would be eager to learn and use" (Wing 2006, p. 33). This argument has been recurring, in one form or another, over the last half century. Perlis (1962), the first recipient of the ACM

Turing Award, said that all undergraduates should learn to program as part of a liberal education (as cited in Guzdial 2008). Papert (1972, 1980) extended this vision to a full literacy starting in childhood. Papert (1996) was the first to use the term computational thinking to refer to the affordances of computational representations for expressing powerful ideas. Similar calls have regularly been made in the decades since (diSessa 2000; Kay and Goldberg 1977; Guzdial and Soloway 2003; Papert 1972, 1980; Wilensky 2001).

The earliest work to put this idea into practice was the development of the Logo programming language (Feurzeig et al. 2011; Papert 1980). While Logo was designed most immediately to teach mathematical concepts, its creators quickly recognized the far reaching benefits of the skills learned through programming, arguing that "computer presence could contribute to mental processes not only instrumentally but in more essential, conceptual ways, influencing how people think even when they are far removed from physical contact with a computer" (Papert 1980, p. 4). Another important perspective on computational thinking comes from the work of diSessa (2000) and his book *Changing Minds*. In particular, diSessa argues that computers can be the basis for a powerful new form of literacy that has the potential to be pervasive across subjects, contexts, and domains.

A number of recent definitions have been proposed for computational thinking without a consensus emerging (Grover and Pea 2013). Wing proposes a definition that emphasizes the unique contribution of the field of computer science to a broad range of human endeavor: "computational thinking involves solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science" (Wing 2006, p. 33). The Royal Society echoes this emphasis on computer science, defining computational thinking as "the process of recognizing aspects of computation in the world that surrounds us, and applying tools and techniques from computer science to understand and reason about both natural and artificial systems and processes" (Furber 2012, p. 29). Highlighting the diversity and inclusive nature of the debate around computational thinking, in 2010, the National Research Council convened a meeting on the scope and nature of computational thinking, producing a report that listed over 20 high-level skills and practices that computational thinking might include, such as problem abstraction and decomposition, heuristic reasoning, search strategies, and knowledge of computer science concepts like parallel processing, machine learning, and recursion (NRC 2010). In this work, we bring a different approach to defining computational thinking by relying on the application of the practices identified above in contexts distinct from computer science. In doing so, we move away from relying on decontextualized ideas and practices and instead draw on real-world instantiations of computational thinking in the wild to provide clarity and specificity on what the term means. Doing so reinforces the argument that these practices are broadly applicable while providing a concrete, actionable definition of computational thinking.

Much of the literature on computational thinking focuses on educational outcomes, including a second meeting convened by the National Research Council focusing on the pedagogical aspects of computational thinking. This effort sought to answer questions such as how computational thinking relates to existing subjects, what a computational thinking progression might look like, how to train teachers in computational thinking practices, and how best to assess computational thinking (NRC 2011b). The Computer Science Teachers Association (CSTA) asserts that "the study of computational thinking enables all students to better conceptualize, analyze, and solve complex problems by selecting and applying appropriate strategies and tools, both virtually and in the real world" (CSTA 2011, p. 9). Although considerable effort has been put into advancing our understanding of computational thinking, there are still challenges to address, particularly in terms of bringing computational thinking into schools. These challenges include defining a learning progression and curriculum, assessing student achievement, preparing teachers, and ensuring equitable access (Grover and Pea 2013). For progress to be made in these areas, it will be necessary to break computational thinking down into a set of well-defined and measurable skills, concepts, and/or practices.

Computational Thinking in K-12 Education

Extensive research over the last three decades has focused on issues related to teaching and learning skills, concepts, and practices relevant to computational thinking (Grover and Pea 2013). There have been a few notable efforts towards creating frameworks and guidelines for bringing computational thinking into K-12 education. Barr and Stephenson (2011), in reporting on work from the computer science education community, provide one approach by proposing a definition for computational thinking across all of K-12 education. As part of this effort, they present a list of computational thinking concepts and map them onto a variety of conventional school subjects, showing, for example, what abstraction could look like in a social studies classroom, or how to use automation in a mathematics lesson. A second effort to provide useful structure for operationalizing computational thinking in K-12 education comes from Brennan and Resnick (2012) who propose a computational thinking framework consisting of three dimensions: computational concepts, computational

practices, and computational perspectives. For each dimension, they articulate what it looks like to engage in computational thinking at that level, and provide guidelines on how to assess computational thinking across the diverse ways it can be used. In parallel to the effort of creating frameworks for understanding and evaluating computational thinking, is the ongoing creation of new learning environments, tools, and activities designed to promote computational competencies in K-12 learning contexts. Such efforts include graphical programming environments such as Scratch (Resnick et al. 2009) and Alice (Cooper et al. 2000); computational modeling environments like STELLA (Richmond et al. 1987) and NetLogo (Wilensky 1999b); electronic prototyping kits such as Arduino and digital textiles (Buechley et al. 2008); video games including Quest Atlantis (Barab et al. 2005) and Robo-Builder (Weintrop and Wilensky 2013); and scaffolded scientific inquiry environments like WISE (Linn et al. 2003), Genscope (Horwitz et al. 1998), GasLab (Wilensky 1999a); Frog Pond—Evolution (Horn et al. 2014), and WorldWatcher (Edelson et al. 1999). Other work focuses not on the technology or medium used for computational thinking instruction, but on the domain in which it is to be situated. Computational thinking practices have been integrated into subjects including history, language arts, mathematics, art, and science (Blikstein and Wilensky 2009; Eisenberg 2002; Hambrusch et al. 2009; Rubin and Nemirovsky 1991; Sengupta et al. 2013; Settle et al. 2012, 2013). A complementary approach to bringing computational thinking in K-12 education is to integrate it in the coursework of pre-service teachers independent of their content specialization (Yadav et al. 2011). The growth of this type of work, the variety of forms it takes, and the diversity of contexts it has been used in, speaks to the need for the cross disciplinary computational thinking framework we present herein.

Many researchers have made the argument that the ability to effectively use computer simulations and interactive visualizations is an important aspect of computational thinking, particularly as it relates to the STEM fields (NRC 2011b). For example, NetLogo (Wilensky 1999b) has successfully been used in schools to introduce students to complex systems and emergent phenomena in many different fields such as probability and statistics (Abrahamson et al. 2006; Abrahamson and Wilensky 2005), chemical reactions (Stieff and Wilensky 2003; Levy and Wilensky 2009), kinetic molecular theory (Brady et al. 2015; Wilensky 1999a), population biology (Wilensky and Reisman 2006), and evolution (Wilensky and Novak 2010; Wagh and Wilensky 2014). Another example is Concord Consortium's Molecular Workbench, which is an interactive modeling platform that enables students to study the motion of particles and provides a simulation platform for teaching and learning science through atomic-scale reasoning (Tinker and Xie 2008). The Physics Education Technology (PhET) project is another example of such learning environments, which provides a large collection of web-based models and simulations for teaching STEM content (Perkins et al. 2006; Bryan 2006).

Another notable approach to bringing computational thinking into K-12 classrooms is the use of online computational resources to enable learning experiences that are otherwise not possible. For example, the iLab Network provides experimental facilities via remote online laboratories that enable students and educators to use real instruments, rather than simulations, and to carry out experiments (Jona and Vondracek 2013). This gives students access to a wide variety of scientific phenomena and control of sophisticated experimental equipment. Activities include studying radioactivity by taking measurements of radioactive material with a Geiger counter, and studying neutron diffraction using a crystal monochromator. Projects such as these bring together science, technology, and computational thinking practices in an accessible and engaging way.

The Growing Role of Computation in Mathematics and Science

The landscape for science is changing. Recent advances in high-speed computation and analytical methods have created powerful tools for understanding phenomena across all spectra of human inquiry. In some scientific fields, such as molecular biology and chemistry, the advent has been recent but rapid. The 1998 Nobel Prize in Chemistry was awarded to John A. Pople and Walter Kohn for their innovative work in the development of computational methods in quantum chemistry (Pople 2003; Kohn 2003). Such a prestigious award hailed the acceptance of computation as a valid and rigorous tool for investigating chemical phenomena. Across a wide variety of domains, the application of statistical and mathematical approaches that rely on computation, such as Markov Chain Monte Carlo methods and artificial neural networks, have proved essential for opening new avenues of exploration and yielded advances in numerous fields as diverse as studying the origins of the universe in computational astrophysics (Vogelsberger et al. 2014) to understanding the kinetics of grain growth in material sciences (Anderson et al. 1984; Srolovitz et al. 1984).

Wing (2006) stated that computational thinking approaches would become fundamental across all disciplines and that the advances in computing would allow researchers to envision new problem solving strategies and to test new solutions in both the virtual and real world. While certain fields such as physics and engineering have a long-standing

interdisciplinary partnership with computational methodologies, classical approaches to problem solving in biology and chemistry have historically emphasized deterministic systems of low complexity, thereby largely ignoring stochastic and nonlinear problems. The previous strong bias toward the study of deterministic systems was primarily one of practicality, with the term "nonlinear" being nearly synonymously with "unsolvable". However, nature is inherently nonlinear and may be characterized by chaotic behavior, as can be observed in climate change (Dijkstra 2013; Manabe and Stouffer 1988), the spread of disease (Keeling and Grenfell 1997; Olsen and Schaffer 1990), ecological distress (Lubchenco et al. 1991), and evolution (Lander and Schork 1994; Turelli and Barton 1994). Both mathematically and physically, deterministic/linear systems are the exceptions rather than the rule. In recent years, computational methods have expanded the range of nonlinear phenomena that can be explored through the use of mathematical and simulation models. Wolfram (2002) even proclaimed the emergence of a new kind of science based on his computational experiments into emergent patterns in nature, arguing such explorations are not possible without computation. Scientific fields are undergoing a renaissance in experimental approaches primarily due to the availability of more powerful computers, accessibility of new analytical methods, and the development of highly detailed computational models in which a diverse array of components and mechanisms can be incorporated. These advances have, in turn, created a growing need to educate students in computational methods and techniques to support the rapidly changing landscape of research across mathematics and scientific disciplines.

Methods

To develop our taxonomy of computational thinking practices for mathematics and science, we drew on multiple resources to identify the characteristic practices that are most important to both meet the needs of students and to reflect the work of professionals across a range of mathematics and science disciplines. Figure 1 illustrates the steps that we followed to create the taxonomy. Throughout the process, we worked closely with other STEM researchers, teachers, and curriculum developers. We primarily drew on three resources for creation and validation of our taxonomy: (1) exemplary educational activities involving computational thinking in mathematics and science, (2) existing concept inventories and standards documents, and (3) interviews with mathematicians and scientists.

Step 1 The first step in the creation of our taxonomy was to review existing computational thinking literature, identifying the skills and practices that are repeatedly cited as being central. This investigation began with the two National Research Council publications on computational thinking (NRC 2010, 2011b) and branched out from there, reviewing literature cited in those reports along with work that builds off the ideas presented in the two reports. Because we see computational thinking as being heavily informed by the fields of engineering and computer science, we also gathered and analyzed computer science, engineering, and technology content frameworks such as the CS Principles project (Astrachan and Briggs 2012), the Computer Science Curricula 2013 report (ACM/IEEE-CS Joint Task Force on Computing Curricula 2013), and the NAEP Technology and Engineering Literacy Framework (Driscoll 2013). In reviewing these documents, our goal was to map out what the existing literature identified as central to computational thinking with a focus on applications to mathematics and science. Throughout this initial step, we sought to understand the broader landscape of computational thinking before narrowing our focus to mathematics and science disciplines. This review produced a preliminary set of ten core computational thinking skills (Table 1), which were used as a starting point for our taxonomy.

Step 2 The second step in creating the taxonomy was to collect and classify a variety of activities designed to introduce computational thinking into high school mathematics and science classrooms. The primary corpus of the activities analyzed was the materials produced as part of "Reach for the Stars," an NSF funded program that links STEM graduate students who use computation in their research with high school teachers to develop classroom-ready activities based on their research.[1] These lessons span a variety of concepts across chemistry, physics, biology, astronomy, earth sciences, networks, and programming. Thirty lesson plans from this collection were selected and coded for elements of computational thinking (12 in physics, 8 in chemistry, 3 in biology, and 7 in mathematics). We also included four lessons from a high school mathematical modeling class designed by a teacher collaborating on the project.

Two researchers independently reviewed each of the 34 activities, looking for specific practices that seemed relevant to computational thinking based on our findings from step 1. This resulted in a combined set of 208 computational thinking "facets" from our activity corpus. Table 2 provides a short excerpt of a portion of the coding of one of our lessons to serve as an example of this process. The lesson being analyzed is a physics activity that uses a virtual roller coaster to allow students to explore the forces and momentum that govern motion.[2] The activity has students design experiments, gather data, analyze their

---

[1] https://gk12northwestern.wikispaces.com/Lesson+Plans.

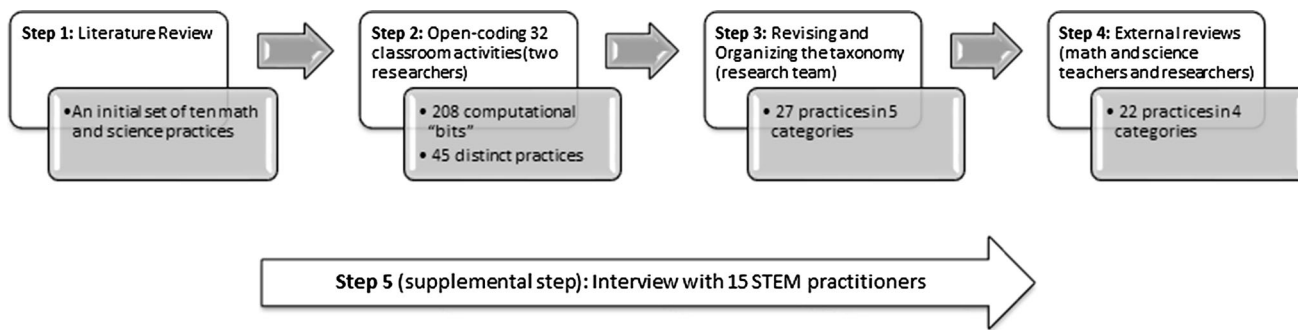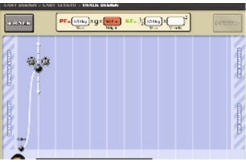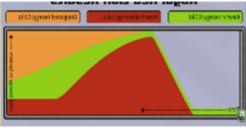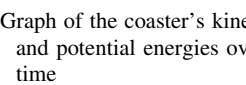[2] https://gk12northwestern.wikispaces.com/Roller+Coaster+Activity.

Fig. 1 Process followed to create the CT in Mathematics and Science Practices taxonomy

Table 1 Initial set of computational thinking skills

Initial set of computational thinking skills	
1. Ability to deal with open-ended problems	6. Creating abstractions for aspects of problem at hand
2. Persistence in working through challenging problems	7. Reframing problem into a recognizable problem
3. Confidence in dealing with complexity	8. Assessing strengths/weaknesses of a representation of data/representational system
4. Representing ideas in computationally meaningful ways	9. Generating algorithmic solutions
5. Breaking down large problems into smaller problems	10. Recognizing and addressing ambiguity in algorithms

Table 2 Example of a portion of the initial coding and final coding for the roller coaster physics lesson

Activity	Computational thinking facet	Computational thinking practice involved	Finalized taxonomy practice
Roller coaster builder	Students build models of roller coasters that can be run, generating data about the coaster's energy	Gain insight/understanding from computer-based simulations/models	Using computational model to understand a concept
			Constructing computational models
Design a roller coaster	Students record energy measurements at four points of the track and store the data in a table	Make effective measurements from a simulation run	Collecting data
	It takes multiple iterations to build a successful roller coaster that finishes the track and does not crash	Iterative approach to a solution	Using computational models to find and test a solution
			Troubleshooting and debugging
Graph of the coaster's kinetic and potential energies over time	Translating on screen graphs of potential/kinetic energy into tabular form	Assessing strengths and weaknesses of a representation	Manipulating data
			Visualizing data

results as they explore the relationship between potential, kinetic, and dissipated energies.

Using the set of 208 computational thinking in mathematics and science facets, two researchers independently open-coded each facet according to the computational thinking practices involved (Column 3 of Table 2). The process started with the set of codes derived from our initial review of the literature (step 1). These skills were often too broad or not tailored for mathematics and science contexts, resulting in the division and refinement of these categories into more specific skills, as well as the introduction of new codes when a facet did not fit within the exiting set of practices. Upon completion of the initial coding, the two researchers iteratively revised the new

codes, resulting in the first full version of our taxonomy, which consisted of 45 distinct computational thinking in mathematics and science practices.

Step 3 The initial set of practices was then shared with the larger research group on the project, which collectively undertook another round of revisions and categorized the individual practices into distinct categories. Much of this work focused on collapsing similar skills into new, unified categories. Also, to obtain external validity, we consulted with the "Reach for the Stars" graduate students who had developed the lesson plans and the in-service teachers who were participating in the project. The resulting product was a revised list of 27 practices thematically grouped into five high-level categories: Data and Information (6 practices), Modeling and Simulation (5 practices), Computation (5 practices), Problem Solving (7 practices), and Systems Thinking (4 practices). Each of these practices was linked with a specific example of it being used in one of the source lesson plans. Table 3 illustrates two examples of these codes and their corresponding computational thinking facet from the lesson plans.

Step 4 The taxonomy was then presented to 16 high school mathematics and science teachers during a summer professional development workshop. As part of the workshop, teachers collaboratively used the taxonomy to design new activities for their classrooms. The feedback from the teachers on the taxonomy was generally positive, but concerns were raised, such as a request we move from the terms *skills* to the broader and more actionable *practices* to reinforce what it means to use these concepts as well as to reflect larger changes in the mathematics and science standards landscape. The teachers were especially wary of the Computation category, which included skills such as applying conditional logic, using recursion and iterative logic, and choosing efficient data structures, as they feared it was too close to computer science and too far from the content they taught in their classrooms.

Along with gathering feedback from teachers, the taxonomy was also presented to computational thinking experts and STEM curriculum designers. These experts raised concerns around the problem solving category, which included practices such as decomposing problems into subproblems and reframing problems into known/familiar problems, as they thought the practices were too general and not unique to STEM or computational thinking contexts. This feedback led to another round of revisions, during which we consolidated the Computation and problem solving categories to address the practical and theoretical concerns raised. The final result of this process was the four-category taxonomy consisting of 22 distinct practices we present in the following section.

Step 5 Throughout the taxonomy generation process, interviews with STEM professionals whose work relies heavily on computation were conducted. Fifteen interviews were carried out with academic faculty from mathematics and science disciplines, STEM researchers in industry, and graduate students pursing degrees in STEM disciplines. This included interviews with biochemists, physicists, material engineers, astrophysicists, computer scientists, and biomedical engineers. The goal of these interviews was to validate the taxonomy and the emerging categories as they were taking shape, as well as to provide supplemental data on the nature of computational thinking as it happens in authentic scientific settings. In analyzing these interviews, we looked for similarities and differences in computational practices across disparate forms of scientific research. From there, we identified and characterized the computational thinking practices that the scientific researchers employed in their work. For example, we found that testing and debugging was a common practice among the scientific researchers that had not been clearly captured in the taxonomy. Through the interviews, we learned that the practice of debugging could be characterized differently for distinct forms of research. For instance, for a theoretical researcher "testing" might involve the process of verifying a solution to an unknown problem, whereas for an experimental researcher, testing might involve the process of computationally validating an experiment before the actual trial. Thus, the description of debugging in the taxonomy needs to reflect this diversity. A full analysis of these data is presented in a forthcoming paper (Beheshti et al. In Preparation).

The Computational Thinking in Mathematics and Science Practices Taxonomy

Our taxonomy is broken down into four major categories: data practices, modeling and simulation practices, computational problem solving practices, and systems thinking practices. Each of these categories is composed of a subset of five to seven practices (Fig. 2). Following the example set by the NGSS, we have chosen to call these "practices" as opposed to "skills" or "concepts" in order "to emphasize that engaging in scientific investigation requires not only skill but also knowledge that is specific to each practice" (NGSS Lead States 2013, p. 30). Although we present our taxonomy as a set of distinct categories, the practices are highly interrelated and dependent on one another. In practice, they are often used in conjunction to achieve specific scientific and mathematical goals. In this section, we describe each of the four major categories, including its constituent practices and our rationale for its inclusion in the taxonomy. In the section that follows the presentation of the taxonomy, we present three classroom activities to demonstrate what the practices look like in use.
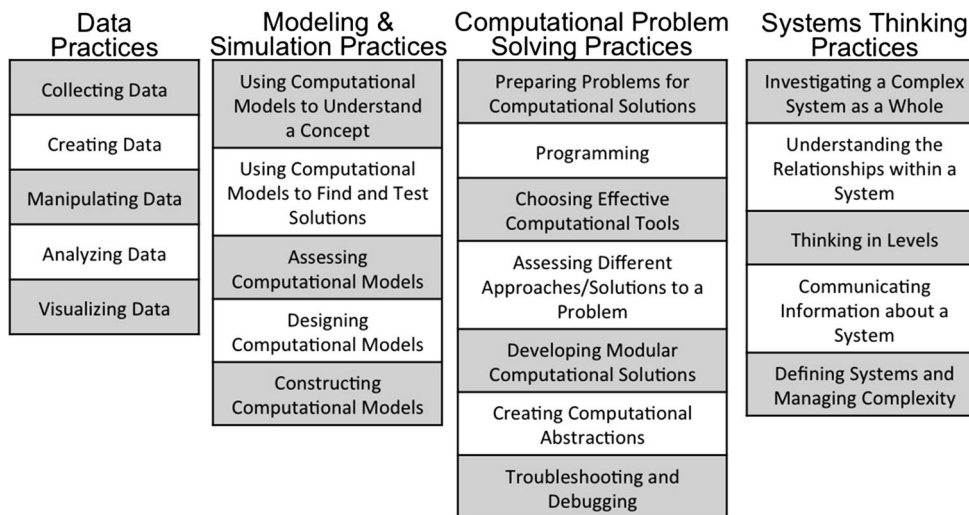
Table 3 Two examples of computational thinking in mathematics and science practices with their associated activities and computational facet

Category	Code	Activity	Computational facet
Data and information	Manipulating data	Network science and hip hop artists[a]: this activity introduces students to network sciences	Students normalize musician's names before entering them into a spreadsheet. Students then sort the data by different criteria to answer questions about the dataset
Modeling and simulation	Assessing model and simulations	Projectile motion lab[b]: this activity has students generate data using the equations that describe motion with Microsoft Excel	Students are asked how valid they think the model is—students identify missing factors like air resistance

[a] https://gk12northwestern.wikispaces.com/Hip+Hop+Networks+Lesson

[b] https://gk12northwestern.wikispaces.com/Projectile+Lab

Fig. 2 Computational thinking in mathematics and science taxonomy

Data Practices
- Collecting Data
- Creating Data
- Manipulating Data
- Analyzing Data
- Visualizing Data

Modeling & Simulation Practices
- Using Computational Models to Understand a Concept
- Using Computational Models to Find and Test Solutions
- Assessing Computational Models
- Designing Computational Models
- Constructing Computational Models

Computational Problem Solving Practices
- Preparing Problems for Computational Solutions
- Programming
- Choosing Effective Computational Tools
- Assessing Different Approaches/Solutions to a Problem
- Developing Modular Computational Solutions
- Creating Computational Abstractions
- Troubleshooting and Debugging

Systems Thinking Practices
- Investigating a Complex System as a Whole
- Understanding the Relationships within a System
- Thinking in Levels
- Communicating Information about a System
- Defining Systems and Managing Complexity

Data Practices

All sciences share certain common features at the core of their problem solving and inquiry approaches. Chief among these is the attitude that data and evidence hold a primary position in deciding any issue (Duschl et al. 2007, p. 27)

Data lie at the heart of scientific and mathematical pursuits. They serve many purposes, take many forms, and play a variety of roles in the conduct of scientific inquiry. The nature of how data are collected, created, analyzed, and shared is rapidly changing primarily due to advancements in computational technologies. "New technologies enable new scientific investigations, allowing scientists to probe realms and handle quantities of data previously inaccessible to them" (NGSS Lead States 2013, p. 32). The importance of being able to use these new technologies to manage and make meaning from the large amounts of data they produce is becoming a defining feature of scientific work in the twenty-first century and thus critical to computational thinking in mathematics and science (Foster 2006). The increasingly computational nature of working with data in scientific and mathematical fields underscores the importance of developing computational thinking data practices in the classroom.

Data skills have long been a part of scientific and mathematical standards and classroom curricula (NCTM 2000; NGSS Lead States 2013). Instruction around the use of data often focuses on developing student understanding of the role of data in investigating questions and using data to construct answers (Hancock et al. 1992; Lehrer and Romberg 1996). Part of the challenge is teaching students that data do not come with inherent structure that lead directly to an answer, but instead that order must be imposed and answers drawn from the data available (Lehrer et al. 2002). The use of data in classrooms also includes introducing the basics of statistics and probability so the data can be used to draw conclusions (Shaughnessy 2007) as well as developing fluency with both conventional and novel data visualizations (diSessa 2004). Increasingly, classrooms have incorporated technology as part of data collection and analysis instruction using tools specifically designed for educational contexts such as Tinkerplots (Konold and Miller 2005), Fathom (Finzer et al. 2001), SimCalc (Roschelle et al. 2000), and NetLogo (Wilensky 1999b) as well as incorporating

standard data analysis tools like Microsoft Excel, SPSS, R, and STATA.

In our analysis of computational thinking lessons for mathematics and science classrooms, data analysis practices were present in 27 of the 34 lessons analyzed. Additionally, the experts we interviewed frequently referenced the importance of computational thinking with respect to the collection, management, and analysis of data in their work. The importance of this category was highlighted in an interview with a materials scientist, who when asked about his research responded: "In almost everything, it's just raw numerical data." He then went on to explain how he computationally defines his research questions, uses supercomputing clusters to generate data, processes and organizes data, and finally uses software packages to generate visualizations of his findings. In every step of his work, computational thinking practices are essential. Below are the five computational thinking practices that comprise the Data category.

Collecting Data

Data are collected through observation and measurement. Computational tools play a key role in gathering and recording a variety of data across many different scientific and mathematical endeavors. Computational tools can be useful in different phases of data collection, including the design of the collection protocol, recording, and storage. Students who have mastered this practice will be able to propose systematic data collection protocols and articulate how those protocols can be automated with computational tools when appropriate.

Creating Data

In many cases, scientists and mathematicians use computational tools to generate data. This is the case when investigating phenomena that cannot be easily observed or measured or that are more theoretical in nature. For example, to understand galaxy evolution, astronomers generate data using computer simulations as it is not possible to observe and measure a galaxy's evolution in situ because the processes occur over billions of years. In this way, computational tools allow for data creation at scales that would otherwise be impossible. Students who have mastered this practice will be able to define computational procedures and run simulations that create data they can use to advance their understanding of the topic under investigation.

Manipulating Data

In mathematical and scientific fields, it is essential to manipulate data in order to make meaning of them. Computational tools make it possible to efficiently and reliably manipulate large and complex datasets. Data manipulation includes sorting, filtering, cleaning, normalizing, and joining disparate datasets. These manipulations serve for both analysis and communication. Students who have mastered this practice will be able to manipulate datasets with computational tools, reshaping the dataset to be in a desired or useful configuration so that it can support further investigation.

Analyzing Data

The true power of data lies in the information that can be gleaned from them through analysis. There are many strategies that can be employed when analyzing data for use in a scientific or mathematical context, including looking for patterns or anomalies, defining rules to categorize data, and identifying trends and correlations. Computational tools have become essential for conducting data analysis, as they make it possible to analyze data in a more reliable, effective manner and to conduct analysis on larger datasets than would otherwise be possible. Using computational tools to analyze data is becoming an especially important practice as we now live in an era of data-intensive science (sometimes referred to as "big data"), where datasets routinely have billions of individual data points. Students who have mastered this practice will be able to analyze a given set of data and make claims and draw conclusions based on the finding from their analysis.

Visualizing Data

Communicating results is an essential component of any knowledge-building endeavor, and computational tools can greatly facilitate that process. In mathematics and science, creating visualizations is a powerful strategy for both analyzing and sharing data. There are a growing number of software tools available for designing and implementing data visualizations (Borner 2015). These tools include both conventional visualizations such as graphs and charts, as well as dynamic, interactive displays that allow the observer to interact with the data being displayed. Students who have mastered this practice will be able to use computational tools to produce visualizations that convey information gathered during analysis.

Modeling and Simulation Practices

> The sciences do not try to explain, they hardly even try to interpret, they mainly make models.
> (von Neumann 1955, p. 628)

The ability to create, refine, and use models of phenomena is a central practice for scientists and

mathematicians (NGSS Lead States 2013). In mathematics and science, models can include flowcharts, diagrams, equations, chemical formulae, computer simulations, and even physical models (Harrison, and Treagust 2000). By definition, models are simplifications of reality that foreground certain features of a phenomenon while approximating or ignoring other features. As such, "all models are wrong, but some are useful" (Box and Draper 1987, p. 424). The *usefulness* of a model comes from its explanatory and/or predictive power and much research has found model-based learning to be an effective pedagogical strategy (for a review, see Louca and Zacharia 2012). Science education is considered to be inseparably intertwined with the development of epistemic and representational practices (Kaput 1998; Lehrer and Schauble 2006; Wilensky and Papert 2010; Wilkerson-Jerde et al. 2015), yet these practices are rarely made explicit as part of instruction. Similar calls have been made of mathematics education. The NGSS and Common Core Mathematics Standards place a new emphasis on not only using models, but also creating models and critically interrogating their limitations and simplifying assumptions.

Computational models and simulations can make scientific concepts more accessible and enhance student understanding of phenomena (Buckley et al. 2004; Klopfer 2003; Parnafes 2007; Schwarz et al. 2007; White and Frederiksen 1998; Wilensky 1997; Wilkerson-Jerde and Wilensky 2015). By computational models, we refer to non-static representations of phenomena that can be simulated by a computer. The pedagogical power of computational models comes not just from students using existing models, but also from enabling students to design, build, and assess models of their own (Brady et al. 2015; Gilbert 2004; Penner 2000; White 1993; Wilensky 1995, 2003; Wilensky and Reisman 2006; Wilkerson-Jerde et al. 2015). Further, computational models make it possible to investigate questions and test hypotheses that would otherwise be too expensive, dangerous, difficult or entirely not possible to carry out (NRC 2011a). As with non-computational models, it is important for students to be able to think critically about computational models to understand their capabilities and limitations.

Of the 34 computational thinking in mathematics and science activities we analyzed, 23 included the use of computational models in various capacities including as tools for problem solving (7) and as tools for exploring concepts (17). Additionally, these activities had learners design, construct, and evaluate models as part of their educational activities. Our modeling and simulation category consists of five computational thinking in mathematics and science practices.

Using Computational Models to Understand a Concept

Computational models that demonstrate specific ideas or phenomena can serve as powerful learning tools. Students can use computational models to deepen their understanding of mathematical and scientific concepts, such as the interdependence within ecosystems, how objects move in a frictionless environment, and probabilistic distributions of random events. Such tools help support the inquiry process by recreating phenomena in environments that support systematic investigation and give the user far more control than would be possible in the natural world. Students who have mastered this practice will be able to advance their own understanding of a concept by interacting with a computational model that demonstrates the concept.

Using Computational Models to Find and Test Solutions

Computational models can also be used to test hypotheses and discover solutions to problems. They make it possible to test many different solutions quickly, easily, and inexpensively before committing to a specific approach. This is especially helpful for phenomena whose outcomes depend on multi-dimensional "parameter spaces." This is an important technique, commonly used when investigating problems in scientific fields and beyond. Students who have mastered this practice will be able to find, test, and justify the use of a particular solution through the use of a computational model as well as be able to apply the information gained through using the model when appropriate.

Assessing Computational Models

A key practice in using a computational model effectively is to understand how the model relates to the phenomenon being represented. This understanding is guided by a variety of questions including: Which aspects of the phenomenon have been faithfully modeled and which aspects have been simplified or ignored? What assumptions have the creators of the model made about the world and how do those assumptions affect its behavior? What layers of abstraction have been built into the model itself and how do these abstractions shape the fidelity of the model? Thinking about these questions is an important part of validating and calibrating a model with respect to the real-world phenomena being represented. Students who have mastered this practice will be able to articulate the similarities and differences between a computational model and the phenomenon that it is modeling, this includes raising issues of threats to validity as well as identifying assumptions built into the model.

Designing Computational Models

Part of taking advantage of computational power in the scientific disciplines is designing new models that can be run on a computational device. The process of designing a model is distinct from actually implementing it; designing a model involves making technological, methodological, and conceptual decisions. There are many reasons that might motivate designing a computational model, including wanting to better understand a phenomenon under investigation, to test out a hypothesis, or to communicate an idea or principle to others in a dynamic, interactive way. When designing a computational model, one is confronted with a large set of decisions including defining the boundaries of the system, deciding what should be included and what can be ignored, and conceptualizing the behaviors and properties of the elements included in the model. Throughout the design process, one must ensure that the resulting model will be able to accomplish the goal that initially motivated the model design process. Students who have mastered this practice will be able to design a computational model, a process that includes defining the components of the model, describing how they interact, deciding what data will be produced by the model, articulating assumptions being made by the proposed model, and understanding what conclusions can be drawn from the model.

Constructing Computational Models

An important practice in scientific and mathematical pursuits is the ability to create new or extend existing computational models. This requires being able to encode the model features in a way that a computer can interpret. Sometimes this takes the form of conventional programming, but in other cases, frameworks and tools support the user in defining behaviors or features through manipulating graphical interfaces or defining sets of rules to be followed. Being able to implement modeling ideas is critical for advancing ideas beyond the work done by others and complements the previous practice of designing computational models. Students who have mastered this practice will be able to implement new model behaviors, either through extending an existing model or by creating a new model either within a given modeling framework or from scratch.

Computational Problem Solving Practice

Applied computer science is now playing the role which mathematics did from the seventeenth through the twentieth centuries: providing an orderly, formal framework and exploratory apparatus for other sciences.

(Djorgovski 2005)

As with much of human endeavor, problem solving is central to scientific and mathematical inquiry. While problem solving can take many forms, in this work, we focus on problem solving practices that are especially effective for working with computational tools and derived from the field of computer science. This perspective builds on work looking at problem solving practices generally (Newell and Simon 1972; Pólya 1954), the power of computational tools to foster the development of such strategies (Clements and Gullo 1984; Palumbo 1990; Papert 1980), and work that brings problem solving and computational tools into scientific domains (Guzdial 1994; Hambrusch et al. 2009; Sengupta et al. 2013; Wilensky 2001; Wilensky et al. 2014).

This category, more so than the others, builds on practices and strategies from the field of computer science, and is intended to capture the field's contribution to contemporary scientific and mathematical work and the importance for today's students to develop this skillset (ACM Pathways 2013; Augustine 2005; Guzdial and Soloway 2003; Henderson et al. 2007). Research has found that enabling students to explore scientific and mathematical phenomena using computational problem solving practices such as programming, algorithm development, and creating computational abstractions can help learners develop deep understandings of mathematical and scientific phenomena (Jackson et al. 1994; Sherin et al. 1993; Taub et al. 2015; Wilensky 1995; Wilkerson-Jerde 2014).

While relatively few of the lesson plans we analyzed would fit into a conventional computer science curriculum, many of the activities include concepts that draw on central practices from the field, including developing algorithms (5 lessons), programming (12 lessons), and working with computational abstractions (8 lessons). Additionally, many of the scientists and mathematicians we interviewed referenced practices from this category including programming, using computational abstractions, and the importance of being able to choose effective computational tools. From this analysis, the resulting Computational problem solving category consists of seven practices intended to lay the groundwork in support of preparing learners for the critical thinking with computation that is becoming central to modern science and mathematics.

Preparing Problems for Computational Solutions

While some problems naturally lend themselves to computational solutions, more often, problems must be reframed so that existing computational tools—be they physical devices or software packages—can be utilized. In the sciences, a vast array of computational tools can be

employed for a given pursuit; the challenge is to map problems onto the capabilities of the tools. Strategies for doing this include decomposing problems into subproblems, reframing new problems into known problems for which computational tools already exist, and simplifying complex problems so the mapping of problem features onto computational solutions is more accessible. Students who have mastered this practice will be able to employ such strategies toward reframing problems into forms that can be solved, or at least progress can be made, through the use of computational tools.

Computer Programming

The ability to encode instructions in such a way that a computer can execute them is a powerful skill for investigating and solving mathematical and scientific problems. Programs ranging from ten-line Python scripts to multi-million-line C++ libraries can be valuable for data collection and analysis, visualizing information, building and extending computational models, and interfacing with other existing computational tools. This practice consists of understanding and modifying programs written by others, as well as composing new programs or scripts from scratch. This category includes understanding programming concepts such as conditional logic, iterative logic, and recursion as well as creating abstractions such as subroutines and data structures. While it is not reasonable to expect all students to be programming experts, basic programming proficiency is an important component of twenty-first century scientific inquiry. Students who have mastered this practice will be able to understand, modify, and create computer programs and use these skills to advance their own scientific and mathematical pursuits.

Choosing Effective Computational Tools

A single task can often be solved a number of different ways using a variety of different computational tools. In such cases, there is often a single tool, or at least a small subset of tools, for the job. Being able to identify the strengths and weaknesses of various possible tools for the problem at hand can be the most important decision in a project. Choosing an effective computational tool includes considering the functionality it provides, its scope and customizability, the type of data the tools expects and can produce, as well as questions that extend beyond the software itself, such as, whether or not there is an active user community that could assist with difficulties you might encounter. Students who have mastered this practice will be able to articulate the pros and cons of using various computational tools and be able to make an informed, justifiable decision.

Assessing Different Approaches/Solutions to a Problem

When there are multiple approaches to solving a problem or multiple solutions to choose from, it is important to be able to assess the options and make an informed decision about which route to follow. This practice is distinct from the previous practice in that it concerns how computational tools, once chosen, will be used, and how they fit in with the larger process of approaching and solving problems. This is important in science and mathematics, as there is often more than one possible course of action. Even if two different approaches produce the same, correct result, there are other dimensions that should be considered when choosing a solution or approach, such as cost, time, durability, extendibility, reusability, and flexibility. Students who have mastered this practice will be able to assess different approaches/solutions to a problem based on the requirements and constraints of the problem and the available resources and tools.

Developing Modular Computational Solutions

When working toward a specific scientific or mathematical outcome, there are often a number of steps or components involved in the process; these steps, in turn, can be broken down in a variety of ways that impact their ability to be easily reused, repurposed, and debugged. Elements of a solution can be large, complicated and uniquely designed for the problem at hand, or they can be small, modular, and reusable. Developing computational solutions in a modular, reusable way has many implications for both the immediate problem and future problems that may be encountered. By developing modular solutions, it is easier to incrementally construct solutions, test components independently, and increase the likelihood that components will be useful for future problems. Students who have mastered this practice will be able to develop solutions that consist of modular, reusable components and take advantage of the modularity of their solution in both working on the current problem and reusing pieces of previous solutions when confronting new challenges.

Creating Computational Abstractions

Creating an abstraction requires the ability to conceptualize and then represent an idea or a process in more general terms by foregrounding the important aspects of the idea while backgrounding less important features. The ability to create and use abstractions is used constantly across mathematical and scientific undertakings, be it creating computational abstractions when writing a program, generating visualizations of data to communicate an idea or

finding, defining the scope or scale of a problem, or creating models to further explore or understand a given phenomenon. Creating computational abstractions is essential for solving multiple problems that have structural similarity but differ in surface detail. These practices are one central way computational power can be brought to bear on mathematic and scientific problems. Students who have mastered this practice will be able to identify, create, and use computational abstractions as they work toward scientific and mathematical goals.

Troubleshooting and Debugging

Troubleshooting broadly refers to the process of figuring out why something is not working or behaving as expected. There are a number of strategies one can employ while troubleshooting a problem, including clearly identifying the issue, systematically testing the system to isolate the source of the error, and reproducing the problem so that potential solutions can be tested reliably. In computer science, this activity is often referred to as "debugging," and there are a number of strategies and tools designed specifically to help with figuring out why a program or other computational tool is not behaving as expected. In STEM fields, the ability to troubleshoot a problem is important, as unexpected outcomes and incorrect behavior are frequently encountered, especially when working with computational tools. Students who have mastered this practice will be able to identify, isolate, reproduce, and ultimately correct unexpected problems encountered when working on a problem, and do so in a systematic, efficient manner.

Systems Thinking Practices

> Instead of looking at one thing at a time, and noting its behavior when exposed to one other thing, the sciences now look at a number of different and interacting things and note their behavior as a whole under diverse influences.
> (Laszlo 1996, p. 4)

So many of the important problems that we face today are complex, involving multiple variables, numerous direct and indirect effects, and are comprised of many, interdependent parts. "The ability to think systemically is an important habit of mind that supports not only the scientific background of the developing STEM workforce, but also future scientifically literate citizens. In a global society where future large-scale, scientifically based decisions will need to be made, it is important for the general populous to develop a systems thinking orientation toward the world" (Duschl and Bismack 2013, p. 120). The systems thinking

approach is fundamentally different from traditional forms of problem solving and focuses on understanding how systems change over time (Forrester 1968). The systems thinking approach has had two distinct foci, with one focus on aggregate systems dynamics (Forrester 1968; Sterman 2000) and the other on agent-based dynamics (Epstein and Axtell 1996; Grimm et al. 2005; Wilensky and Resnick 1999; Wilensky and Rand 2015). Both kinds of systems exhibit emergent behaviors (Bar-Yam 2003; Jacobson and Wilensky 2006; Sterman 2000; Wilensky and Resnick 1999). With the emergence of so many challenges related to big data, many systems thinking methods have become critical components of computational thinking approaches practiced across scientific disciplines. Traditional analyses focus on breaking down problems into their constituent parts and looking separately at the individual pieces. Systems thinking analyses, in contrast, focus on an inclusive examination of how the system and its constituent parts interact and relate to one another as a whole (Assaraf and Orion 2005).

Many fundamental (and difficult) scientific concepts are best, and perhaps only, understood through a systems lens. Good examples include natural selection and population dynamics in ecology, the human respiratory system from biology, and the ideal gas laws in chemistry and physics (Hmelo et al. 2000; Stieff and Wilensky 2003; Wilensky and Reisman 2006). Other types of systems, such as those encountered in physics, economics, and even history, involve cross-cutting concepts related to systems thinking such as feedback, emergence, stocks, and flows (Goldstone and Wilensky 2008; Penner 2000; Wilensky and Rand 2015; Zuckerman and Resnick 2003). Recommendations for the systems perspective are now represented in four separate NRC reports—*Taking Science to School* (NRC 2007), *A Framework for K-12 Science Education* (NRC 2012a), *Education for Life and Work* (NRC 2012c) and *Discipline-Based Education Research* (NRC 2012b) as well as the *Next Generation Science Standards* (NGSS Lead States 2013). While this knowledge is not necessarily bound to computational thinking, computational tools enable new ways to explore, understand, and represent these ideas (Bar-Yam 2003; Sterman 2000; Wolfram 2002). Computation therefore serves as a powerful medium to make these ideas accessible to learners (Klopfer 2003; Wilensky 2001).

In our analysis of computational thinking activities designed for high school mathematics and science classes, eight lessons included developing practices associated with systems thinking. These findings were reinforced by the interviews we conducted with scientists and mathematicians, as they frequently cited concepts from complex systems and practices associated with investigating and understanding systems through computation as being important in their field. While many of the computational

practices that we describe above are valuable for investigating systems, this section introduces five computational thinking practices that focus on systems thinking.

Investigating a Complex System as a Whole

A system can be viewed as a single entity composed of many interrelated elements; for some questions, it is more effective to investigate how the system works as a whole as opposed to studying each individual element or set of elements. Investigating a complex system as a whole relies on the ability to define and measure inputs and outputs of the system. This is especially critical in the sciences because so many phenomena are the result of very large-scale, complex interactions. Being able to black box the details of the underlying systematic interactions and focus on the system as a whole makes it possible to understand the characteristics of the system in aggregate, which is sometimes exactly the data needed for the problem at hand. Computational tools such as models and simulations are especially useful in such investigations, as they can automate pieces of an investigation, take precise measurements, and model systems for further analysis and hypothesis testing. Another powerful approach to facilitate students in learning about complex systems as whole can come from the use of representations (including feedback, stocks and flows, agents, and agent rules) that depict systems in a nonlinear fashion. Students who have mastered this practice will be able to pose questions about, design and carry out investigations on, and ultimately interpret and make sense of, the data gathered about a system as a single entity.

Understanding the Relationships within a System

Whereas some questions can best be answered by focusing on a system as a whole, other questions require understanding how the components within a system interact. Thus, it is important to be able to identify the different elements of a system and articulate the nature of their interactions. Computational tools are useful for conducting such inquiry as they can provide learners with controls for isolating different elements, investigating their behaviors, and exploring how they interact with other components of the system. Students who have mastered this practice will be able to identify the constituent elements of a system, articulate their behaviors, and explain how interactions between elements produce the characteristic behaviors of the system.

Thinking in Levels

Systems can be understood and analyzed from different perspectives, ranging from a micro-level view that considers the smallest elements of the system to a macro-level view that considers the system as a whole. Thinking about a system from the standpoint of its individual actors and components can lead to insights about how micro-level behaviors lead to emergent macro-level patterns. On the other hand, being able to black box the details of the underlying systematic interactions and focus on the system as a whole makes it possible to understand the emergent characteristics of the system in aggregate, which can yield a different set of insights from a micro-level analysis. Computational tools can facilitate the investigation of the system from both perspectives, and, as Levy and Wilensky (2008) show, from productive mid-levels between the two. Students who have mastered this practice will be able to identify different levels of a given system, articulate the behavior of each level with respect to the system as a whole, and be able to move back and forth between levels, correctly attributing features of the system to the appropriate level.

Communicating Information about a System

A central challenge when investigating a system is figuring out how best to communicate what you have learned about it. This is often challenging due to the size and complexity of the system under investigation. Because systems often consist of many interrelated parts and can include numerous interacting elements, conveying information about the system can be difficult, but is also essential for the information gleaned from the system to be understood and used by others. Communicating information about a system often involves developing effective and accessible visualizations and infographics[3] that highlight the most important aspects of what has been learned about the system in such a way that it can be understood by someone who does not know all the underlying details. This practice includes the ability to prioritize features of a system, design intuitive ways to represent it, and identify what can be left out of the visualization without compromising the information being conveyed. Students who have mastered this practice will be able to communicate information they have learned about a system in a way that makes the information accessible to viewers who do not know the exact details of the system from which the information was drawn.

---

[3] By infographic, we mean a visual abstraction that communicates information. It includes conventional formats such as graphs, charts, and maps, but also includes interactive, dynamic visualizations designed with the express goal of communicating information to the viewer.

Defining Systems and Managing Complexity

Anything can be viewed as a system; the size and membership of the system depends on where you define its boundaries. You can have very small systems that include only a narrow set of entities, like a classroom system consisting of a teacher and students, or you can have systems that include millions of entities, like a group of galaxies or the human genome. The larger the system is, in terms of number of entities, types of entities, frequency of interactions, and diversity of behavior, the more complex it becomes. The decision of where to set the boundaries of the system is critical for any investigation that follows as it determines what questions you can answer as well as the size and complexity of the system. Further, in order to leverage computational tools in working with systems, one must explicitly delineate the boundaries of the systems that are under investigation. It is important to be able to define a system in a way that is useful and productive. This includes creating a system that includes all the necessary elements to be able to accomplish the desired goal while limiting its size, complexity, and scope. Students who have mastered this practice will be able to define the boundaries of a system so that they can then use the resulting system as a domain for investigating a specific question as well as to identify ways to simplify an existing system without compromising its ability to be used for a specified purpose.

Computational Thinking in Mathematics and Science in Practice

As part of our effort to bring computational thinking into mathematics and science classrooms, we are developing a series of computational thinking enhanced lesson plans for high school mathematics, biology, chemistry, and physics classrooms. These activities are informed by our taxonomy, introduced in professional development workshops, and taught by teachers without formal backgrounds in computational thinking or computer science. The activities have been designed to be easily adopted and incorporated into existing mathematics and science curricula. Here we present three lessons that demonstrate the computational thinking in mathematics and science (CT-MS) practices outlined in our taxonomy.[4]

Video Games: Physics Phact or Phiction?

This activity is designed for high school physics classrooms and has students investigate the laws of physics that govern video games. Depending on the selected game,

students calculate the gravitational constant (e.g., in Angry Birds) and/or conservation of energy and momentum (e.g., in Asteroids). The lesson begins with students creating a 20- to 30-s screencast of themselves playing the game, then use Physics Tracker (Brown 2013), a video transcription and computational analysis tool, to collect data (CT-MS Practice: Data Collection). Within Physics Tracker, the students visualize the data, plotting time, position, velocity, acceleration, and other germane physical parameters (CT-MS Practice: Data Visualization). They then use Tracker's analysis tools to identify the best fit between the data and their mathematical functional form (CT-MS Practice: Data Analysis). Students derive the properties that define the behavior of the world, such as finding the gravitational constant in Angry Birds (CT-MS Practice: Using Computational Models to Find/Test Solutions). Having answered this question, students assess the video game as a model, identify differences with the real world, and communicate why those design choices help the usability of the game (CT-MS Practices: Assessing Computational Models, Communicating Information about a System).

DNA Sequencing from the Ground Up

This activity is designed for high school biology classrooms and has students imagine that they are scientists in the 1990s faced with the task of sequencing the entire human genome. Students start with a simple case of putting back together a verse of a familiar song that has been broken into pieces and randomly arranged. After reflecting on this puzzle and the strategies that they developed to reassemble the lyrics (CT-MS Practice: Assessing Different Approach/Solutions to a Problem), students go on to a more difficult challenge: reconstructing an unknown password with an unfamiliar combination of letters and numbers. They then apply their technique from the password puzzle to derive an efficient, robust, and general algorithm for sequencing things (CT-MS Practices: Preparing Problems for Computational Solutions, Creating Computational Abstractions). After determining their technique for assembling the full DNA sequence, the students write their procedure in pseudocode, intended to be the first step toward expressing the algorithm in a manner that a computer could interpret and execute (CT-MS Practice: Programming). Classmates then swap algorithms, attempt to implement them, and provide suggestions for improvement in terms of clarity and efficiency (CT-MS Practice: Assessing Different Approaches/Solutions to a Problem). At the end of the lesson, the students are asked to consider why computers were so essential in accomplishing the task of sequencing the human genome. By having them think about and experience scaling the problem up from a simple case to a more complex problem, the students develop an

---

[4] Additional information on the lesson plans, including materials and software, can be found at http://ct-stem.northwestern.edu

appreciation for the processing speed that computers provide and how computation has revolutionized the questions we ask and the way we do science.

Gas Laws

In this high school chemistry lesson, students use an interactive simulation to explore the relationships between macroscopic properties of gases—pressure, volume, and temperature—based on how these properties emerge from microscopic interactions (i.e., the motion of the gas particles and the interaction with the walls of the container). The tool is part of the PhET suite of simulations (Adams et al. 2008a, b) that has been found to help students build conceptual understandings of the concepts being modeled (CT-MS Practice: Using Computational Models to Understand a Concept) and is based on the earlier GasLab modeling toolkit (Wilensky 1999a). Students explore the behaviors of gases by running experiments, the result of which are stored in a spreadsheet (CT-MS Practice: Creating Data). The students then visualize their data, graphing it in various formats to find the most compelling way to explain the relationship between the variables to classmates (CT-MS Practices: Analyzing Data, Visualizing Data). The simulation allows the students to investigate both the microscopic, agent-level behaviors and macroscopic, aggregate properties of gases (CT-MS Practice: Thinking in Levels). At the end of the lesson, students are asked whether there are aspects of the simulation that are unrealistic or different from the real world. Students are also asked to reflect on the difference between exploring the gas laws with models versus trying to conduct table experiments (CT-MS Practice: Assessing Computational Models). To derive the gas laws themselves, the students must consider the system as a whole, backgrounding the details of the underlying interactions and focusing on identifying and measuring inputs to and outputs from the aggregate system (CT-MS Practice: Investigating a Complex System as a Whole).

Conclusion

This paper argues for the inclusion of computational thinking in mathematics and science classrooms. We see three main benefits for the approach of embedding computational thinking in these contexts: (1) it builds on the reciprocal relationship for learning between computational thinking and mathematics and science domains, (2) it addresses practical concerns of reaching all students, and having proficient teachers, and (3) it brings science and mathematics education more in line with current professional practices in these fields. We have presented a taxonomy that articulates a definition of "computational thinking in mathematics and science" and contributes a language around which standards, curricula, and assessments can develop. In so doing, we address the issue of ambiguity and a lack of precision that has plagued much of the discussion surrounding computational thinking. We see the work presented in this paper as a first step in the process of bringing computational thinking into mathematics and science classrooms. Achieving this goal requires the support of a diverse set of stakeholders to be successful. This includes teachers becoming comfortable teaching the material and receiving professional development in computation-based lessons and technology; school administrators allocating resources to support its inclusion; policy makers prioritizing computational thinking as a part of mathematics and science education; curriculum and assessment developers producing computational thinking materials targeted for science and mathematics classrooms; and the broader community supporting the effort to bring computational thinking into these educational spaces. The inclusion of computational thinking as a core scientific practice in the Next Generation Science Standards and similar language in mathematics' standards are important milestones, but there is still much work to do toward addressing the challenge of educating a technologically and scientifically savvy population and preparing the next generation of world class scientists.

References

Abrahamson D, Wilensky U (2005) ProbLab goes to school: design, teaching, and learning of probability with multi-agent interactive computer models. In: Proceedings of the fourth conference of the European Society for research in mathematics education. San Feliu de Gixols

Abrahamson D, Janusz RM, Wilensky U (2006) There once was a 9-block: a middle-school design for probability and statistics. J Stat Educ 14(1). http://www.amstat.org/publications/jse/v14n1/abrahamson.html

Adams WK, Reid S, LeMaster R, McKagan SB, Perkins KK, Dubson M, Wieman CE (2008a) A study of educational simulations part I: engagement and Learning. J Interact Learn Res 19(3):367–419

Adams WK, Reid S, LeMaster R, McKagan SB, Perkins KK, Dubson M, Wieman CE (2008b) A study of educational simulations part II: interface design. J Interact Learn Res 19(4):551–557

Anderson MP, Srolovitz DJ, Grest GS, Sahni PS (1984) Computer simulation of grain growth—I: kinetics. Acta Metall 32(5):783–791

Assaraf OB-Z, Orion N (2005) Development of system thinking skills in the context of earth system education. J Res Sci Teach 42(5):518–560

Astrachan O, Briggs A (2012) The CS principles project. CM Inroads 3(2):38–42

Augustine NR (2005) Rising above the gathering storm: energizing and employing America for a brighter economic future. National Academies Press, Washington, DC

Bailey D, Borwein JM (2011) Exploratory experimentation and computation. Not Am Math Soc 58(10):1410–1419

Barab S, Thomas M, Dodge T, Carteaux R, Tuzun H (2005) Making learning fun: Quest Atlantis, a game without guns. Educ Technol Res Dev 53(1):86–107

Barr V, Stephenson C (2011) Bringing computational thinking to K-12: what is Involved and what is the role of the computer science education community? ACM Inroads 2(1):48–54

Bar-Yam Y (2003) Dynamics of complex systems. Perseus Publishing, New York

Beheshti E, Weintrop D, Horn MS, Orton K, Jona K, Wilensky U (In Preparation) Computational thinking in the wild: how scientists and mathematicians use computational thinking in their work.

Blikstein P (2013) Digital fabrication and "making" in education: the democratization of invention. In: Walter-Herrmann J, Büching C (eds) FabLabs: of machines, makers and inventors, Transcript Publishers, Bielefeld, pp 1–21

Blikstein P, Wilensky U (2009) An atom is known by the company it keeps: a constructionist learning environment for materials science using agent-based modeling. Int J Comput Math Learn 14(2):81–119

Borner K (2015) Atlas of knowledge: anyone can map. MIT Press, Cambridge

Box GE, Draper NR (1987) Empirical model-building and response surfaces. Wiley, New York

Brady C, Holbert N, Soylu F, Novak M, Wilensky U (2015) Sandboxes for model-based inquiry. J Sci Educ Technol 24(2):265–286

Brown D (2013) Tracker: video analysis and modeling tool (Version 4.82). http://www.cabrillo.edu/~dbrown/tracker

Brennan K, Resnick M (2012). New frameworks for studying and assessing the development of computational thinking. Presented at the American Education Researcher Association, Vancouver, Canada.

Bryan J (2006) Technology for physics instruction. Contemp Issues Technol Teach Educ 6(2):230

Buckley BC, Gobert JD, Kindfield ACH, Horwitz P, Tinker RF, Gerlits B et al (2004) Model-based teaching and learning with BioLogica^{TM}: what do they learn? how do they learn? how do we know? J Sci Educ Technol 13(1):23–41

Buechley L, Eisenberg M, Catchen J, Crockett A (2008) The LilyPad Arduino: using computational textiles to investigate engagement, aesthetics, and diversity in computer science education. In: Proceedings of the SIGCHI conference on Human factors in computing systems. ACM, New York, pp 423–432

Chinn CA, Malhotra BA (2002) Epistemologically authentic inquiry in schools: a theoretical framework for evaluating inquiry tasks. Sci Educ 86(2):175–218

Clements DH, Gullo DF (1984) Effects of computer programming on young children's cognition. J Educ Psychol 76(6):1051

Computer Science Teachers Association (2011) K-12 computer science standards. http://csta.acm.org/Curriculum/sub/K12Standards.html

Confrey J (1993) The role of technology in reconceptualizing functions and algebra. Paper presented at the 17th Annual Meeting of the North American Chapter of the International Group for the Psychology of Mathematics Education, Asilomar

Cooper S, Dann W, Pausch R (2000) Alice: a 3-D tool for introductory programming concepts. J Comput Sci Coll 15(5):107–116

Dijkstra HA (2013) Nonlinear climate dynamics. Cambridge University Press, Cambridge

diSessa AA (2000) Changing minds: computers, learning, and literacy. MIT Press, Cambridge

diSessa AA (2004) Metarepresentation: native competence and targets for instruction. Cogn Instr 22(3):293–331

Driscoll DP (2013) Technology and engineering literacy framework for the 2014 National Assessment of Educational Progress. US Department of Education, Washington DC

Duschl RA, Bismack AS (2013) Standards for science education: quantitative reasoning and modeling concepts. In: Duschl RA, Bismack AS (eds) Reconceptualizing STEM education: the central role of practices. University of Wyoming, Laramie, WY

Duschl RA, Schweingruber HA, Shouse AW (2007) Taking science to school: learning and teaching science in grades K-8. National Academies Press, Washington, DC

Edelson DC, Gordin DN, Pea RD (1999) Addressing the challenges of inquiry-based learning through technology and curriculum design. J Learn Sci 8(3/4):391–450

Eisenberg M (2002) Output devices, computation, and the future of mathematical crafts. Int J Comput Math Learn 7(1):1–44

Epstein J, Axtell R (1996) Growing artifical societies: social science from the bottom up. Brookings Institution Press, Washington

Feurzeig W, Papert S, Lawler B (2011) Programming-languages as a conceptual framework for teaching mathematics. Interact Learn Environ 19(5):487–501

Finzer W, Erickson T, Binker J (2001) Fathom [computer software]. KCP Technologies, Emeryville

Forrester JW (1968) Principles of systems. Pegasus Communications, Waltham, MA

Foster I (2006) 2020 computing: a two-way street to science's future. Nature 440(7083):419

Furber S (2012) Shut down or restart? The way forward for computing in UK schools. Technical report, The Royal Society, London

Gardner DP (1983) A nation at risk: the imperative for educational reform. U.S. Department of Education, Washington, DC

Gilbert JK (2004) Models and modelling: routes to more authentic science education. Int J Sci Math Educ 2(2):115–130

Goldstone RL, Wilensky U (2008) Promoting transfer by grounding complex systems principles. J Learn Sci 17(4):465–516

Google: Exploring Computational Thinking. (n.d.). Retrieved 25 Oct 2010. http://www.google.com/edu/computational-thinking/index.html

Grimm Volker, Revilla Eloy, Berger Uta, Jeltsch Florian, Mooij Wolf M, Railsback Steven F, Thulke Hans-Hermann, Weiner Jacob, Wiegand Thorsten, DeAngelis Donald L (2005) Pattern-oriented modeling of agent-based complex systems: lessons from ecology. Science 310:987–991

Grover S, Pea R (2013) Computational thinking in K-12: a review of the state of the field. Educ Res 42(1):38–43

Guzdial M (1994) Software-realized scaffolding to facilitate programming for science learning. Interact Learn Environ 4(1):001–044

Guzdial M (2008) Paving the way for computational thinking. Commun ACM 51(8):25–27

Guzdial M, Soloway E (2003) Computer science is more important than calculus: the challenge of living up to our potential. SIGCSE Bull 35(2):5–8

Hambrusch S, Hoffmann C, Korb JT, Haugan M, Hosking AL (2009) A multidisciplinary approach towards computational thinking for science majors. In: ACM SIGCSE bulletin, vol 41, pp 183–187

Hancock C, Kaput JJ, Goldsmith LT (1992) Authentic inquiry with data: critical barriers to classroom implementation. Educ Psychol 27(3):337

Harrison AG, Treagust DF (2000) A typology of school science models. Int J Sci Educ 22(9):1011–1026

Henderson PB, Cortina TJ, Wing JM (2007) Computational thinking. In: ACM SIGCSE bulletin, vol 39. ACM, pp 195–196

Hmelo CE, Holton DL, Kolodner JL (2000) Designing to learn about complex systems. J Learn Sci 9(3):247–298. doi:10.1207/S15327809JLS0903_2

Horn MS, Brady C, Hjorth A, Wagh A, Wilensky U (2014) Frog pond: a code-first learning environment on evolution and natural selection. ACM Press, New York, pp 357–360

Horwitz P, Schwartz J, Kindfield ACH, Yessis LM, Hickey DT, Heidenberg A, Wolfe EW (1998) Implementation and evaluation of the GenScope^{TM} learning environment: issues, solutions, and results. In: Guzdial M, Kolodner J, Bruckman A (eds) Proceedings of the 3rd annual international conference of the learning sciences. Association for the Advancement of Computers in Education, Charlottesville

Jackson SL, Stratford SJ, Krajcik J, Soloway E (1994) Making dynamic modeling accessible to precollege science students. Interact Learn Environ 4(3):233–257

Jacobson MJ, Wilensky U (2006) Complex systems in education: scientific and educational importance and implications for the learning sciences. J Learn Sci 15(1):11–34

Jona K, Vondracek M (2013) A remote radioactivity experiment. Phys Teach 51(1):25

Jona K, Wilensky U, Trouille L, Horn MS, Orton K, Weintrop D, Beheshti E (2014) Embedding computational thinking in science, technology, engineering, and math (CT-STEM). Presented at the Future Directions in Computer Science Education Summit Meeting, Orlando

Kaput JJ (1998) Representations, inscriptions, descriptions and learning: a kaleidoscope of windows. J Math Behav 17(2):265–281

Kay A, Goldberg A (1977) Personal dynamic media. Computer 10(3):31–41

Keeling MJ, Grenfell BT (1997) Disease extinction and community size: modeling the persistence of measles. Sci 275(5296):65–67

Klopfer E (2003) Technologies to support the creation of complex systems models: using StarLogo software with students. Biosystems 71(1–2):111–122

Kohn W (2003) Nobel lectures, chemistry 1996–2000. World Scientific Publishing Co, Singapore, p 213

Konold C, Miller CD (2005) TinkerPlots: dynamic data exploration. Computer software. Key Curriculum Press, Emeryville, CA

Lander ES, Schork NJ (1994) Genetic dissection of complex traits. Sci 265(5181):2037–2048

Laszlo E (1996) The systems view of the world: a holistic vision for our time, 2nd edn. Hampton Press, Cresskill, NJ

Lehrer R, Giles N, Schauble L (2002) Data modeling. In: Lehrer R, Schauble L (eds) Investigating real data in the classroom: expanding children's understanding of mathematics and science. Teachers College Press, New York, pp 1–26

Lehrer R, Romberg T (1996) Exploring children's data modeling. Cognition Instruct 14(1):69–108

Lubchenco J, Olson AM, Brubaker LB, Carpenter SR, Holland MM, Hubbell SP et al (1991) The sustainable biosphere initiative: an ecological research agenda: a report from the Ecological Society of America. Ecology 72(2):371–412

Lehrer R, Schauble L (2006) Cultivating model-based reasoning in science education. In: Sawyer RK (ed) The Cambridge handbook of the learning sciences. Cambridge University Press, New York, pp 371–388

Levy ST, Wilensky U (2008) Inventing a "Mid Level" to make ends meet: reasoning between the levels of complexity. Cogn Instr 26(1):1–47

Levy ST, Wilensky U (2009) Crossing levels and representations: the connected chemistry (CC1) curriculum. J Sci Educ Technol 18(3):224–242

Lin CC, Zhang M, Beck B, Olsen G (2009) Embedding computer science concepts in K-12 science curricula. In: Proceedings of the 40th ACM technical symposium on computer science education. ACM, New York, pp 539–543

Linn MC, Clark D, Slotta JD (2003) WISE design for knowledge integration. Sci Educ 87(4):517–538

Louca LT, Zacharia ZC (2012) Modeling-based learning in science education: cognitive, metacognitive, social, material and epistemological contributions. Educ Rev 64(4):471–492

Manabe S, Stouffer RJ (1988) Two stable equilibria of a coupled ocean–atmosphere model. J Clim 1:841–866

Margolis J (2008) Stuck in the shallow end: education, race, and computing. The MIT Press, Cambridge

Margolis J, Fisher A (2003) Unlocking the clubhouse: women in computing. The MIT Press, Cambridge

National Governors Association Center for Best Practices, Council of Chief State School Officers (2010) Common core state standards for mathematics. National Governors Association Center for Best Practices, Council of Chief State School Officers, Washington, DC

National Research Council (2007) Taking science to school: learning and teaching science in grades K-8. National Academies Press, Washington, DC

National Research Council (2010) Report of a workshop on the scope and nature of computational thinking. The National Academies Press, Washington, DC

National Research Council (2011a) Learning science through computer games and simulations. The National Academies Press, Washington, DC

National Research Council (2011b) Report of a workshop of pedagogical aspects of computational thinking. The National Academies Press, Washington, DC

National Research Council (2012a) A framework for K-12 science education: practices, crosscutting concepts, and core ideas. National Academies Press, Washington, DC

National Research Council (2012b) Discipline-based education research: understanding and improving learning in undergraduate science and engineering. National Academies Press, Washington, DC

National Research Council (2012c) Education for life and work: developing transferable knowledge and skills in the 21st century. National Academies Press, Washington, DC

Newell A, Simon HA (1972) Human problem solving, vol 104. Prentice-Hall, Englewood Cliffs, NJ

NGSS Lead States (2013) Next generation science standards: for states, by states. The National Academies Press, Washington, DC

Olsen LF, Schaffer WM (1990) Chaos versus noisy periodicity: alternative hypotheses for childhood epidemics. Sci 249(4968):499–504

Palumbo DB (1990) Programming language/problem solving research: a review of relevant issues. Rev Educ Res 60(1):65–89

Papert S (1972) Teaching children to be mathematicians versus teaching about mathematics. Int J Math Educ Sci Technol 3(3):249–262

Papert S (1980) Mindstorms: children, computers, and powerful ideas. Basic books, New York

Papert S (1996) An exploration in the space of mathematics educations. Int J Comput Math Learn 1(1):138–142

Parnafes O (2007) What does "fast" mean? understanding the physical world through computational representations. J Learn Sci 16(3):415–450

Penner DE (2000) Cognition, computers, and synthetic science: building knowledge and meaning through modeling. Rev Res Educ 25:1

Perkins K, Adams W, Dubson M, Finkelstein N, Reid S, Wieman C, LeMaster R (2006) PhET: interactive simulations for teaching and learning physics. Phys Teach 44(1):18

Perlis A (1962) The computer in the university. In: Greenberger M (ed) Computers and the world of the future. MIT Press, Cambridge, pp 180–219

Pople J (2003) Nobel lectures, chemistry 1996–2000. World Scientific Publishing Co, Singapore, p 246

Redish EF, Wilson JM (1993) Student programming in the introductory physics course: mUPPET. Am J Phys 61:222–232

Repenning A, Webb D, Ioannidou A (2010) Scalable game design and the development of a checklist for getting computational thinking into public schools. In Proceedings of the 41st ACM technical symposium on computer science education. pp 265–269

Resnick M, Silverman B, Kafai Y, Maloney J, Monroy-Hernández A, Rusk N et al (2009) Scratch: programming for all. Commun ACM 52(11):60

Richmond B, Peterson S, Vescuso P, Maville N (1987) An Academic user's guide to Stella Software. High Performance Systems, Inc, Lyme, NH

Roschelle J, Kaput J, Stroup W (2000) SimCalc: accelerating student engagement with the mathematics of change. In: Learning the sciences of the 21st century: research, design, and implementing advanced technology learning environments. pp 47–75

Rubin A, Nemirovsky R (1991) Cars, computers, and air pumps: thoughts on the roles of physical and computer models in learning the central concepts of calculus. In Underhill RG (ed) Proceedings of the Thirteenth International Conference for the Psychology of Mathematics Education—North American Chapter Conference, Virginia, pp 168–174

Ryoo JJ, Margolis J, Lee CH, Sandoval CD, Goode J (2013) Democratizing computer science knowledge: transforming the face of computer science through public high school education. Learn Media Technol 38(2):161–181

Schwarz CV, Meyer K, Sharma A (2007) Technology, pedagogy, and epistemology: opportunities and challenges of using computer modeling and simulation tools in elementary science methods. J Sci Teach Educ 18(2):243–269

Sengupta P, Kinnebrew JS, Basu S, Biswas G, Clark D (2013) Integrating computational thinking with K-12 science education using agent-based computation: a theoretical framework. Educ Inf Technol 18(2):351–380

Settle A, Franke B, Hansen R, Spaltro F, Jurisson C, Rennert-May C, Wildeman B (2012) Infusing computational thinking into the middle- and high-school curriculum. In: Proceedings of the 17th ACM conference on Innovation and technology in computer science education. ACM, New York, pp 22–27

Settle A, Goldberg DS, Barr V (2013) Beyond computer science: computational thinking across disciplines. In: Proceedings of the 18th ACM conference on innovation and technology in computer science education. ACM, New York, pp 311–312

Shaughnessy JM (2007) Research on statistics learning. In: Lester FK (ed) Second handbook of research on mathematics teaching and learning. Information Age Publishing, Charlotte, NC, pp 957–1009.

Sherin BL (2001) A comparison of programming languages and algebraic notation as expressive languages for physics. Int J Comput Math Learn 6(1):1–61

Sherin BL, diSessa AA, Hammer D (1993) Dynaturtle revisited: learning physics through collaborative design of a computer model. Interact Learn Environ 3(2):91–118

Srolovitz DJ, Anderson MP, Sahni PS, Grest GS (1984) Computer simulation of grain growth—II: grain size distribution, topology, and local dynamics. Acta Metall 32(5):793–802

Sterman J (2000) Business dynamics: systems thinking for a complex world. Irwin/McGraw-Hill, New York

Stieff M, Wilensky U (2003) Connected chemistry: incorporating interactive simulations into the chemistry classroom. J Sci Educ Technol 12(3):285–302

Taub R, Armoni M, Bagno E, Ben-Ari M (2015) The effect of computer science on physics learning in a computational science environment. Comput Educ 87:10–23

Tinker RF, Xie Q (2008) Applying computational science to education: the molecular workbench paradigm. Comput Sci Eng 10(5):24–27

Turelli M, Barton NH (1994) Genetic and statistical analyses of strong selection on polygenic traits: what, me normal? Genetics 138(3):913–941

Vogelsberger M, Genel S, Springel V, Torrey P, Sijacki D, Xu D et al (2014) Introducing the illustris project: simulating the coevolution of dark and visible matter in the Universe. Mon Not R Astron Soc 444(2):1518–1547

von Neumann J (1955) Method in the physical sciences. In: Bródy F, Vámos T (eds) The Neumann compendium: world series in 20th century mathematics, vol 1. World Scientific Publishing Co, Singapore, p 628

Wagh A, Wilensky U (2014) Seeing patterns of change: supporting student noticing in building models of natural selection. In: Proceedings of 2014 constructionism. Vienna, 19–23 Aug

Weintrop D, Wilensky U (2013) RoboBuilder: a computational thinking game. In Proceeding of the 44th ACM technical symposium on computer science education. ACM, Denver, pp 736–736

White BY (1993) ThinkerTools: causal models, conceptual change, and science education. Cogn Instr 10(1):1

White BY, Frederiksen JR (1998) Inquiry, modeling, and metacognition: making science accessible to all students. Cogn Instr 16(1):3–118

Wilensky U (1995) Paradox, programming, and learning probability: a case study in a connected mathematics framework. J Math Behav 14(2):253–280

Wilensky U (1997) What is normal anyway? therapy for epistemological anxiety. Educ Stud Math 33(2):171–202

Wilensky U (1999a) GasLab: an extensible modeling toolkit for exploring statistical mechanics. In: Roberts N, Feurzeig W, Hunter B (eds) Computer modeling and simulation in science education. Springer, Berlin, pp 151–178

Wilensky U (1999b) NetLogo. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston. http://ccl.northwestern.edu/netlogo

Wilensky U (2001) Modeling nature's emergent patterns with multi-agent languages. In: Proceedings of EuroLogo. Linz, pp 1–6

Wilensky U (2003) Statistical mechanics for secondary school: the GasLab multi-agent modeling toolkit. Int J Comput Math Learn 8(1):1–41

Wilensky U, Novak M (2010) Teaching and learning evolution as an emergent process: the BEAGLE project. In: Taylor R, Ferrari M (eds) Epistemology and science education: understanding the evolution versus intelligent design controversy. Routledge, New York

Wilensky U, Papert S (2010) Restructurations: reformulations of knowledge disciplines through new representational forms. In: Clayson J, Kalas I (eds) Proceedings of the constructionism 2010 conference. Paris. 10–14 Aug, p 97

Wilensky U, Rand W (2015) An introduction to agent-based modeling: modeling natural, social and engineered complex systems with NetLogo. MIT Press, Cambridge

Wilensky U, Reisman K (2006) Thinking like a wolf, a sheep, or a firefly: learning biology through constructing and testing computational theories—an embodied modeling approach. Cogn Instr 24(2):171–209

Wilensky U, Resnick M (1999) Thinking in levels: a dynamic systems approach to making sense of the world. J Sci Educ Technol 8(1):3–19

Wilensky U, Brady C, Horn M (2014) Fostering computational literacy in science classrooms. Commun ACM 57(8):17–21

Wilkerson-Jerde MH (2014) Construction, categorization, and consensus: student generated computational artifacts as a context for disciplinary reflection. Educ Technol Res Dev 62(1):99–121

Wilkerson-Jerde MH, Wilensky U (2015) Patterns, probabilities, and people: making sense of quantitative change in complex systems. J Learn Sci 24(2):204–251

Wilkerson-Jerde MH, Gravel BE, Macrander CA (2015) Exploring shifts in middle school learners' modeling activity while generating drawings, animations, and computational simulations of molecular diffusion. J Sci Educ Technol 24(2–3):396–415

Wing JM (2006) Computational thinking. Commun ACM 49(3):33–35

Wolfram S (2002) A new kind of science, 1st edn. Wolfram Media, Tokyo

Yadav A, Zhou N, Mayfield C, Hambrusch S, Korb JT (2011) Introducing computational thinking in education courses. In: Proceedings of the 42nd ACM technical symposium on Computer science education, ACM, pp 465–470

Zuckerman O, Resnick M (2003) System blocks: a physical interface for system dynamics learning. In: Proceedings of the 21st international system dynamics conference. Citeseer, pp 810–811.