3 Computing Education

Literature Review and Voices from the Field

Paulo Blikstein and Sepi Hejazi Moghadam

Dedicated to Francisco Walter Durán Segarra (in memoriam), Ecuadorian polymath, professor, and educational researcher, an ahead-of-his-time mind who never gave up the fight for a more emancipatory, meaningful, and democratic education.

3.1 Introduction

In 1967, Seymour Papert, Cynthia Solomon, and Wally Feurzeig created the Logo computer language, the first designed for children (Papert, 1980) – an event widely considered as the beginning of computing education (CEd).[1] In a time when computers cost millions of dollars and occupied entire rooms, teaching computing to children, while visionary, was a hard sell for school systems and policy-makers. From the mid-1970s to the early 1990s, CEd slowly penetrated schools worldwide. Despite a decade of popularity in the 1980s, it never reached as deeply into the educational mainstream as Papert and his colleagues wished. Since the mid-2000s, however, there has been a pronounced shift in the focus on science, technology, engineering, and mathematics (STEM) education, and CEd is at the forefront of this process (National Research Council, 2012). As computational technologies have become inexpensive and pervasive in our lives, the demand for an educated and technologically literate labor force has continued to increase (Noonan, 2017; US Department of Labor, 2007). This is not merely about the labor force, but also about citizenship. The need for children to become future producers of technology – fluent in the medium of our time, instead of merely consumers – has become a major focus for policy-makers and researchers. Today, educators and CEd advocates are pushing ahead with plans to add computing to the list of topics that all students should study (K–12 Computer Science Framework Steering Committee, 2016).

Other catalysts driving the mainstream acceptance of CEd include the launch of the Scratch, Blockly, NetLogo, and Alice programming environments; the launch of organizations such as the Computer Science Teachers Association (CSTA; an international body founded by the Association for Computing Machinery [ACM]), the rise of the maker movement and fablabs (Blikstein, 2013, 2018); the creation of organizations providing CS learning opportunities

1 John Kemeny and Thomas Kurtz (Dartmouth College) created the BASIC programming language in 1964, but Logo is used as a landmark because of its comprehensive focus on all segments and age levels of education, especially children.

such as Code.org, Black Girls Code, Girls Who Code, and others[2]; and the rollout of national programs such as CS4All in the USA. As a result, there is an almost overwhelming demand from school systems worldwide for research and implementation guidelines, one that the relatively small CEd research community is simply not able to meet (Guzdial, 2017). The newness of the discipline is also an important factor. For example, while the US National Council of Teachers of Mathematics (NCTM) was founded in 1920, and its science counterpart, the National Science Teachers Association (NSTA), was formed in 1944, CSTA was not launched until 2004. When NCTM and NSTA were formed, school infrastructure was already in place for these disciplines, thousands of mathematics and science teachers were teaching in schools across the USA, and teacher colleges supported a strong pipeline for more. *CEd does not have those advantages today.* The current focus on CEd has also generated much discourse regarding its purpose. Is the rationale for CEd to fulfill job market needs, promote personal empowerment, teach children to code, develop students' fluency in a new literacy, address historical educational inequalities, or some combination of all of the above?

The goal of this chapter (which is based on a longer report originally commissioned by Google[3,4]) is to better understand the state of CEd by using a methodological innovation. Instead of only examining the published literature, we also interviewed some of the main voices in the field, inquiring about two topics in particular: the multiple rationales for teaching computing and the obstacles for sustainable implementation. With these goals in mind, this chapter summarizes interviews conducted with several leading researchers and practitioners, in addition to providing an examination of literature reviews and articles.

3.2 Methods

We utilized three main data sources for this chapter: interviews, literature reviews, and analysis of papers recommended by the interviewees. For the interviews, we selected leaders in the field from various universities, institutions, and organizations, trying to balance intellectual traditions, academic backgrounds, and expertise. The selection focused mostly on the USA, not because it is representative of what happens in other countries, but mostly to have a more complete picture of CEd in one country. The final group of interviewees consisted of 14 people: Matthew Berland (University of Wisconsin-Madison), Leah Buechley (Rural Digital), Michael Clancy (University of California,

2 There is a large number of such organizations, many focusing on underserved populations: Black Girls Code, Girls Who Code, Girls Code it, CoderDojo, Technovation, and Yes We Code.

3 Available at https://services.google.com/fh/files/misc/pre-college-computer-science-education-report.pdf

4 Stanford University has strict rules to avoid conflicts of interest or bias in reports written for private entities. The original report was created following those rules.

Berkeley), Andrea “Andy” diSessa (University of California, Berkeley), Sally Fincher (University of Kent), Shuchi Grover (formerly SRI International), Mark Guzdial (Georgia Institute of Technology), Mike Horn (Northwestern University), Jane Margolis (University of California, Los Angeles), Mitchel Resnick (Massachusetts Institute of Technology), Sue Sentance (King’s College, London), Ben Shapiro (University of Colorado, Boulder), David Weintrop (University of Maryland), and Pat Yongpradit (Code.org).

All invited interviewees accepted to be interviewed, except one professor, who nominated another scholar in his own department (Michael Clancy, University of California, Berkeley), and Andrea diSessa, who preferred to send an in-preparation paper instead (the paper is used in this chapter in lieu of an interview and is listed in the references). All participants were given the option of anonymity and none opted for it. After the first complete draft was finished, all 14 interviewees were given the opportunity to fully review the text and suggest further changes, which were individually considered for the final version.

We used a semi-structured protocol for the interviews that included questions about the relevance and importance of teaching computing, the main research findings in the field, and research, policy, and implementation agendas for years to come. The interviews were conducted remotely by the first author via videoconference, audio recorded, transcribed in their entirety, and analyzed using a grounded coding approach. The principal themes extracted from the initial coding were: (a) teacher preparation; (b) policy and scale-up; (c) curriculum development; (d) cultural, diversity, and equity issues; (e) pedagogy; and (f) historical aspects of CEd. These categories informed a further refining of the coding, so the data were recoded for more fine-grained topics, resulting in approximately 1,000 excerpts grouped into 130 sub-codes. Those were then recategorized in terms of the six initial themes and informed the structure of the analysis. For the purposes of this chapter, we will focus mostly on two of those six main clusters of: the rationales for teaching computing and CEd implementation.

The literature was selected using a combination of recommendations from the interviewees, well-established policy documents such as the CSTA K–12 Computer Science Standards (Seehorn et al., 2011) and the K–12 Computer Science Framework (K–12 Computer Science Framework Steering Committee, 2016), foundational works in the field, and existing literature reviews. We used the literature to add a layer of peer-reviewed research to the topics extracted from the interviews, and triangulated research findings across interviews and the literature.

We chose this hybrid format (interviews and reviews) to simultaneously capture well-established facts and findings, but also novel information that has not yet made it to the publication venues in the field. Also, some of the important challenges and issues in CEd often do not show up in peer-reviewed publications because many active members of the community are also tool developers instead of researchers – so their work could not be entirely captured in a traditional literature review. This combined use of interviews and literature offered a more comprehensive view of the state of the very young and dynamic field of CEd.

We will reference the interviews using the conventional reference format for personal communication (it might help readers to keep in mind that most references from 2017 are, in fact, the interviews).

3.3 Findings: Rationales for Justifying CS Education

The first theme emerging from the interviews and literature, and one of the main topics of this chapter, was the differing reasons for teaching CS and their considerable consequences for CS implementation programs. Similar to a recent study by Vogel, Santo, and Ching (2017), we found that the interdisciplinary nature of CS brings together very different stakeholders and views. CEd includes professionals from different academic cultures and professional allegiances: university professors, K–12 educators, CEOs of technology companies, entrepreneurs, government officials, and diversity and equity advocates. Not surprisingly, the data from the interviews and literature revealed many different justifications for why CS should be taught in public education systems (e.g., diSessa, 2000; Wing, 2006). These rationales can be expressed as four distinct positions:

- *The labor market rationale:* Labor market changes and the need to sustain a competitive economy are the main driving forces for this rationale. Some consider that CS knowledge will be useful not only for professional programmers but also in a variety of twenty-first-century non-technical jobs, thus universally valuable for all professions.
- *The computational thinking rationale:* The argument for "computational thinking" is that computer scientists' ways of thinking, heuristics, and problem-solving strategies are universally important, and would transfer to a variety of knowledge domains and everyday problems. It would also support the development of students' higher-order thinking skills.
- *The computational literacy rationale:* Computational literacy is not a new skill or a class of problem-solving strategies, but a set of material, cognitive, and social elements that generate new ways of thinking and learning. It enables new types of mental operations and knowledge representations, creates new kinds of "literatures," makes it possible for people to express themselves in new ways, changing how people accomplish cognitive tasks.
- *The equity of participation rationale:* CS knowledge will be required for the best and most creative jobs, for civic participation, and for understanding the impact of computation on society. Additionally, since our cognitive capabilities will be limited by our ability to utilize computation, equity of participation in CEd becomes the central concern, and is one of the most significant gaps in research and implementation.

Making these four rationales explicit is important because they drive the way we write curricula, train teachers, and implement CEd in schools. Interviewees pointed out that the public's lack of awareness about these different viewpoints – and the ways they are similar, dissimilar, complementary, and compatible – must be addressed (e.g., Buechley, 2017; Resnick, 2017).

3.3.1 The Labor Market Rationale

Changes in the global labor market have been a major driver of the efforts to teach CS in schools. This rationale is primarily related to the demands for more workers with new skill sets and is frequently championed by industry leaders and policy-makers. The labor market argument comes in two chief forms. The first cites the hundreds of thousands of open jobs in CS (Google LLC & Gallup Inc., 2016; Grover & Pea, 2013) and notes that this number will increase in years to come, with data science and artificial intelligence becoming mainstream fields relevant across many industries. Similarly, it is argued that the economic productivity or contributions of a country will be determined by its capacity to generate more scientists and engineers. CEd can presumably contribute to this vision by fixing the "leaky" STEM pipeline and driving more students to pursue CS careers. However, Grover and Horn point out that in grades K–8 especially, this concern with jobs might be misplaced:

> In elementary school, students and teachers are definitely not thinking about jobs. It is about what are the foundational knowledge and skills that children should have? At the middle school level, even though it is not a jobs argument, I think there is an identity argument there. This is especially relevant to computing because there are so many stereotypes associated with it.
> (Grover, 2017)

> We have gone a little too far on the commercial end of the spectrum, we have become preoccupied with training the next generation of engineers, these economic motivations are outweighing the computational literacy ideas.
> (Horn, 2017)

The second form the labor market argument takes is a subtler one. It argues for more CS knowledge embedded in all careers, instead of simply training more programmers. Several of the interviewees mentioned that while professional programmers will be necessary, the need could be restricted to a relatively small number of positions that are highly specialized (Guzdial, 2017; Resnick, 2017; Shapiro, 2017). Some reports suggest that only about 6 percent of the workforce will need to do coding with the scope and specialization of professional programmers (Noonan, 2017). The greatest demand would not be for professional programmers, but for other professionals who will have to use CS and programming for automating spreadsheets, programming queries, accessing online databases, using data-mining software tools, and operating physical computing devices in interactive art or home automation.

3.3.2 The Computational Thinking Rationale

The second argument for teaching CS derives from the concept of "computational thinking" (CT), as put forth in a position paper written by Jeanette Wing (2006). Wing proposed that computer scientists' ways of thinking, heuristics, and problem-solving strategies are universally important both for applying computing ideas to do work in other disciplines and for applying computing ideas in everyday life. Examples are the ability to use abstractions and pattern recognition to represent problems in new ways, to break down problems into smaller parts, and to employ algorithmic thinking. With 3,500 citations (according to Google Scholar as of April 2018), the position put forward by Wing has played a critical role in shaping the world of CEd. Her paper and her influential position as a National Science Foundation (NSF) officer helped reinvigorate the field. Some researchers, however, are skeptical about how well students transfer CS knowledge to everyday life and general problem-solving. diSessa (2018) mentions that there have been several attempts over the last 100 years to teach children transferable problem-solving or higher-order thinking skills (HOTS) using mathematics, Latin, or Greek, but these endeavors often failed. Guzdial (2017) mentions several studies on the transfer of CEd knowledge and points out that generally "students fail to apply even simple computing ideas to fairly simple problems." Yongpradit further notes that:

> CEd is not immune to the misconceptions about high-level transfer. I know that there are advocates … saying that computing can improve general critical thinking skills. That's not supported by research. It will not magically improve your math scores.
> (Yongpradit, 2017)

Because Wing's original ideas are still influential in the field, the need for more empirical evidence and the absence of a more definitive unpacking of the term CT are considered to be major issues in CEd – after all, would the "ways of thinking" of computer scientists transfer to other domains and contexts?

However, the definition of CT has been evolving over the last few years, and steering away from the original one put forth by Wing, as Grover notes:

> The definition of CT has been evolving since Wing, and in its evolution it has broadened to encompass aspects of CT concepts, practices, as well as learners' dispositions and perspectives, perhaps fueled by a genuine desire to broaden participation, thus including aspects such as creativity, collaboration, and communication in practices of CT.
> (Grover, 2017)

CT is further discussed in Chapters 17–20 of this Handbook.

3.3.3 The Computational Literacy Rationale

With more than 1,100 citations (according to Google Scholar as of April 2018), Andrea diSessa's book *Changing Minds* is the most established account of the idea of "computational literacy" (diSessa, 2000). In the book, and in recent publications (diSessa, 2018), he explains how different computational literacy is from the original definition of CT (a similar discussion appears in Wilensky & Papert, 2010).

> Learning to use a new medium takes effort. The printing press was a huge leap in human history, but that leap did not happen until many more people became literate. A printing press is not of much use unless authors know how to write and your audience knows how to read. Achieving computational literacy in society means that people can read and write with computation, which includes an ability to read and write computer programs.
> (diSessa, 2000)

> I view computation as, potentially, providing a new, deep, and profoundly influential literacy – computational literacy – that will impact all STEM disciplines at their very core, but most especially in terms of learning.
> (diSessa, 2018)

diSessa claims that computational literacy is not simply a new job skill or generic CS-inspired problem-solving strategy, but a set of material, cognitive, and social elements that generate a new way of knowing, thinking, learning, and representing knowledge. A new literacy makes new types of mental operations and knowledge representations possible, creates new kinds of previously nonexistent "literatures," and changes how people interact with each other and use digital devices when they are accomplishing cognitive tasks. He also mentions that there is a semantic confusion between computational literacy versus terms like digital literacy, computer literacy, or information communication and technology (ICT) literacy. These latter terms refer to the competent use of different computational devices and technologies. Computational literacy, conversely, is concerned with how computational media can change the way we know, learn, and think (in contrast with the focus on problem-solving or HOTS).

diSessa also argues that concepts in science and mathematics can be made simpler using computational representations. For example, velocity and acceleration are simple to understand algorithmically, but unnecessarily complex to learn using traditional algebraic representations. Chemical processes such as diffusion, given their probabilistic nature, are convoluted when represented in algebraic terms, but very simple to learn using computational tools such as agent-based models (e.g., NetLogo; Wilensky, 1999), in which students can program the behavior of individual atoms. The argument for computational literacy extends beyond the need for teaching programming languages. It makes the claim that several disciplines could be fundamentally transformed if taught using computational tools, in the same way that text literacy changed the teaching of so many disciplines centuries ago.[5] Sentance, Resnick, and Horn also stress that computational literacy is multifaceted, and more than just learning CT or programming concepts:

> I think computational thinking skills exist … I think we just have to be careful about thinking that computing is only computational thinking. CS … involves

5 Text literacy fundamentally changed how we accomplish cognitive operations – for example, it acts as external memory, it is shareable, and it is permanent. diSessa and others claim that computational literacy could have the same revolutionary consequences.

> modeling and design and creativity, more than just the cognitive elemental thinking skills. That is what we need to teach in K–8. We need to teach the whole subject and be cautious of being too narrow in what we are offering in the curriculum in school.
> (Sentance, 2017)

> Gaining a literacy is a matter of developing your thinking, your voice, and your identity … The reason for learning to write is not just for doing practical things but being able to express your ideas to others. Computation is a new way of expressing ourselves and it's important for everyone to learn … If you want to feel like a full participant in the culture, you need to be a contributor with the media of the times.
> (Resnick, 2017)

> It is about supporting computation in many different genres or niches. As a poet, the way you use computation might be very different than a journalist, a researcher, or somebody who works in government. Just like we have different forms of literacy, we might have different forms of computational literacy.
> (Horn, 2017)

However, as diSessa states, discussions about the role and importance of CEd are far from over, and these views should all be earnestly considered with their implicit contradictions:

> The labor market view and the computational thinking view contain at least implicit criticisms of the computational literacy view. The former might think that immediate and practical economic effects are more important, and the latter suggests that computational literacy is diffuse, hard to implement, and might insist that high-order thinking skills do exist, so these perspectives should not be ignored.
> (diSessa, 2018)

Some interviewees pointed out that the boundaries between CT and computational literacy are not well-defined. While Grover (2017) states that new definitions of CT have been evolving to include, for example, creativity and collaboration, formerly mostly associated with computational literacy, Guzdial (2017) worries that these new CT definitions "are going too broad," and Resnick notes that the definition of CT "out in the field" is still very much connected to the original one as stated in Wing's (2006) paper. Computational literacy is further discussed in Chapters 18 and 19.

3.3.4 The Equity of Participation Rationale

Several interviewees mentioned equity as their central concern in CEd, arguing that it has traditionally been a side issue in the field and one of the most significant gaps in research and implementation. There are two main issues related to the topic:

- Understanding the impact of computation on society, and
- Ensuring equity and diversity in participation.

The K–12 Computer Science Framework (K–12 Computer Science Framework Steering Committee, 2016) also recognized equity and broadening participation as core issues in CEd. Students excluded from CEd may struggle to fully participate in twenty-first-century society along multiple dimensions. Not only will the best and most creative jobs require CS knowledge, but our cognitive capabilities to solve problems will be limited by our inability to utilize computation fully. Even traditional forms of civic participation will require an understanding of Computing. As Buechley stated:

> We live in a computationally mediated world, and it is important for people to have an understanding of how computational systems work and the role that they play in those systems, how those systems impact their lives, our democracy, the economy, and the way we socialize and interact with people.
> (Buechley, 2017)

Several interviewees gave examples of how computing will become increasingly crucial for civic participation and informed decision-making. These examples include knowing what algorithms are, how computational tools can manipulate social media, how to participate in a social discourse mediated by algorithms, and how to make sense of job displacement due to automation. It is also important to be aware of the presence and consequences of technologies such as machine learning (ML) and artificial intelligence (AI) in a number of everyday devices and experiences, understanding how much information we divulge (sometimes unknowingly) about ourselves, and being aware of the ways in which bias can get built into technologies that influence critical decisions such as prison sentencing, mortgage allocation, and the deployment of neighborhood policing resources (O'Neil, 2016; Shapiro, 2017). The comprehension of the rapidly evolving landscape of devices and tools that are key for active participation in modern society is also central to this argument. Students who do not fully understand these issues risk being more easily manipulated as consumers, voters, and citizens, and more vulnerable to cybercrime. They also are less likely to have access to leadership positions and high-status jobs and are more likely to be on the sidelines of future societal change.

The interviewees also noted that Computing drives innovation across many disciplines and industries and that the resulting changes have had both an economic and a sociological impact. Some also said that allowing students to explore their social and cultural concerns using computing helps motivate and engage them and makes Computing relevant to their lives, especially in diverse populations (Margolis, 2017). Buechley (2017) adds that when you put computing in contexts that can be compelling and exciting to different groups of people, "you get diverse populations to show up and participate," and stresses the importance of making conscious, deliberate space for that to happen. Many interviewees noted that private and more affluent schools will most certainly be able to offer CEd programs with high complexity, while less affluent or public school systems will only offer very simplified versions:

Private schools do not do just generic education. They have kids working on portfolios. They have children doing internships. They have kids doing projects and making it relevant to them … Standardized education which has no connection to kids' lives is what is often given to poor kids.
(Margolis, 2017)

[I was] working first in informal settings and then in recent years, I have moved more in the formal space. I saw it as being more relevant because that is now seen as a way to level the playing field and make sure that all children get it, not just those that happen to be fortunate to get it through after-school experiences.
(Grover, 2017)

Grover noted that the Obama administration's naming of the national CEd effort as "Computer Science for All" when it was announced in January 2016 supported this equity-oriented perspective:

This of course came as a result of notions the community grew to accept over the previous 5 years … CSForAll is now a well-used term that captures this "equity of participation" notion.
(Grover, 2017)

Sentance (2017) stresses the importance of making CS mandatory in all schools, for all students, not as mere "exposure," but as a way to avoid self-selection. The interviewees also noted that the lack of a diverse CS workforce results in the design of products and services that cater to a very narrow range of people and problems, thus perpetuating inequality. Researchers concerned with the equity argument also posit that we could see a much more perverse version of the "digital divide" in the years to come if immediate and intentional actions are not taken to address these inequities while we are still in early design stages of CEd. Equity issues are further discussed in Chapters 16 and 24.

3.4 Sustainable Implementation and Systemic Obstacles

The second cluster of findings stemming from the interviews and the literature relates to key components needed across the CEd system to support wider and more effective implementation. The "system" we define includes the various interrelated institutions and mechanisms that shape and support CEd teaching and learning in the classroom.

The six key components of CEd implementations reviewed in this section are equity, scaling, quality of implementation, pluralism, curriculum, and teacher development. It is difficult to focus on any particular component without considering how it is influenced by – and how it in turn influences – the other components. For example, what students learn is clearly related to what they are taught, which itself depends on many elements: the instructional materials available in the market; the curriculum adopted locally; teachers' content and pedagogical knowledge; how teachers elect to use the curriculum; the kinds of resources, time, and space that teachers have for their practice; what the community values regarding student learning; and how local, state, and national standards and assessments influence instructional practice.

I am not attempting to provide a full discussion of all possible influences on CEd; rather, I focus on the themes that emerged from the data and how they might contribute to a more coherent and inclusive implementation of CEd.

3.4.1 Equity and Broadening Participation

Several interviewees mentioned broadening participation in and changing perceptions of CS as perhaps the most important challenges for our community. Berland, Buechley, Margolis, Sentance, and others stressed the striking contrast between what happens in Computing classrooms in affluent schools and in less affluent schools. Almost all of the interviewees expressed concern with the unequal presence of programming in public schools, the quality of instruction, and the unconscious bias of some educators and counselors regarding who is "suited" to take the Computing classes. They also noted that while affluent schools are more likely to offer comprehensive Computing programs for their students, most public districts are ill-equipped to offer anything more than very brief, standardized experiences, which they fear could give school administrators and teachers an incorrect metric for CEd adoption and distract them from implementing more robust programs in their schools. The interviewees also worry that the numbers of children reached as advertised by nonprofits and industry providers give the impression that the "mission has been accomplished," whereas most agree that we are still very far from providing CEd to all students. At least three researchers also noted that funding currently provided to large national organizations would be better directed to research institutions or smaller, more local nonprofits. But Yongpradit (2017) noted that national organizations can be a channel for funding to smaller organizations.

Regarding broadening participation, most interviewees favored programs that make learning Computing more attractive by focusing on personal expression and creativity, especially at the K–8 level. They also agreed on the importance of culturally relevant curricula that support diverse ways of approaching CEd and diverse ways of expressing one's knowledge. Buechley (2017), for example, mentioned that computer scientists and engineers tend to discount culture and cultural relevance as key factors in learning and in tool design. In her work, she instead focuses on creating *new types of clubhouses* and computing cultures that speak to these diverse practices. Michael Clancy also advocated for CEd that incentivizes meaningful engagement:

> Students will be more motivated to work if the assignments allow creativity, and allow the student to relate to his or her experience. Part of that would be more flexible tools that allow a student to make better use of his or her experience. What I would like to see is some way to have a broader scope and interest of activities.
> (Clancy, 2017)

Some identified the need to make CEd mandatory for all students as a means of ensuring equitable participation. Sentance (2017), for example, argued that "if we don't make computing mandatory, we know from previous experience that self-selecting groups of people will choose computing … so we have a responsibility to offer that to all children and to reach everybody." Yongpradit (2017) stated that schools should at least be required to offer CEd, and that we should make CEd courses available permanently for students in public schools. Guzdial (2017) expressed concern that some states are trying to implement "CS4All" without an explicit focus on underserved groups. He points out that affluent schools will be able to move quickly to provide Computing for their students, while less affluent schools will struggle with financial limitations, further exacerbating the "coding divide." Margolis also noted that, while the "CS for California" campaign has an equity agenda:

> The rush to scale and the pressure to put curriculum and teacher professional development (PD) online will possibly have dangerous unintended consequences for the issue of equity … The learning partnership of teachers and of researchers needs to become part of a dynamic iterative cycle for continuous improvement … For programs to sustain themselves, to change the culture of the schools so that teachers are supported to have active, engaged, inclusive classrooms, for programs to be fully embraced by the districts themselves. It is the slow work of relationship building and learning together that is required. For this to happen there also needs to be a holistic awareness of all the educational issues in schools that continue to threaten equity. CS in schools does not exist on isolated islands. All of the large issues impacting education, such as the move for privatization, de-professionalizing teachers, and school tracking will affect our broadening participation in the computing mission.
> (Margolis, 2017)

3.4.2 Scaling and Assessment

Buechley, Shapiro, Berland, and other interviewees expressed concern about traditional forms of school reform taking over the implementation of CEd. Specifically, they noted that fixed curricula, standardized assessments, and inflexible teacher training programs do not foster real scientific or mathematical thinking in students (National Research Council, 2006, 2012) and have a questionable track record for motivating students to pursue STEM careers (Maltese & Tai, 2011). For Buechley (2017), one dominant narrative around CEd is that "we need to figure out the concepts, and teach them in the right way in a fixed curriculum." She disagreed with this narrative, however, and instead advocated for a perspective in which motivation, engagement, personally relevant projects, and culturally aware curriculum design take precedence. According to Buechley, CEd lends itself especially well to projects and interdisciplinary work that connect programming to art, design, biology, or mathematics:

> Connecting computation and computing to different practices, which sometimes coincide with really different ways of approaching and making sense of the world, is the most powerful way that you can engage different kinds of people in computing … As one example, I have been connecting computation to textile crafts, textile design, and fashion design, and I have found that through doing that, you can dramatically change the gender participation ratios. You

> can get lots of young women to engage enthusiastically with computing in a way that they just do not do in more traditional computing contexts.
>
> Computer science is a fundamentally creative discipline. You construct things when you write a computer program. And in that sense, it's really distinct from mathematics or science. That is a distinction that is not fully appreciated and made sense of, but is very powerful and important.
> (Buechley, 2017)

Berland expressed a similar concern:

> There are very few subjects in which students feel like they can make a change in the world and they can express their independent selves. I think their ability to make their own games, make their own art, make them in ways that are shareable with code, is really powerful. [Instead of giving students the right answer] it is better to create safe spaces to fail, to play, to tinker … This is where you get the bang for the buck. That's where the learning happens. Another truism of education is that things are driven by the ways that they are assessed. If you assess people for knowing this or that keyword in C++,[6] then that's what you're going to get and that's not particularly valuable, but if you assess people on their ability to teach each other complex concepts, that's what you're going to get.
> (Berland, 2017)

Fincher (2017) cited the UK's Project Quantum[7] as an example of an explicitly research-based project that combines scholarly work, practical utility, curriculum scaffolds, and teacher PD.

3.4.3 Reliance on Surface-Level, Low-Quality Solutions

Another topic that was mentioned by many interviewees as a systemic obstacle was related to the pace and depth of many of the current CEd implementations, pointing to the fact that many seem to be superficial and overly simplified, especially in public education. Margolis expressed concerns about the speed at which they are being developed and put into classrooms, and argued that this approach has unintended educational consequences, especially for members of underrepresented groups:

> [The idea of many programs is] ship it out. Get it out there and we will see if there are bugs in it, right? That has some real potential dangers in education because you put something online and the school district says, "Okay, we're going to do computing online," and then all of a sudden the girls and a lot of the students of color don't do well, and then the principal says, "See? Our kids are not up for computing. They didn't do well. They're not interested." In fact, they just experienced horrible instruction, and so they get turned off, but in their minds they're not cut out for it, and in the minds of the principals they're not cut out for it.
> (Margolis, 2017)

Grover voiced a similar concern. She has been observing and researching citywide implementations in the USA and examining the quality and depth of the projects.

6 C++ is a very popular professional programming language.
7 http://community.computingatschool.org.uk/resources/4382/single

She noted the simplicity of the projects she observed and the need to more deeply engage groups that have been historically underrepresented in Computing:

> Almost no one uses Boolean logic. They use variables but just as a count or a score. You barely ever see expressions with variables being used or you will rarely see a loop with a terminating condition that is controlled by a Boolean expression with variables. Also, I read this paper from Yasmin Kafai and Deborah Fields where they analyzed the Scratch community projects [in 2012].[8] Most children stayed at the shallow end, they used the simplest constructs.
> (Grover, 2017)

Shapiro (2017) voiced concerns about the concentration of resources in just a few CEd organizations, which could lead to "very homogeneous curricula/programs which would move us in the opposite direction" from many of the progressive approaches discussed in the CEd community. Similar concerns have been voiced by many prominent educators in light of large-scale implementations in many US cities. As those implementations roll out, the quality of instruction has often been criticized as superficial, stifled, and insufficient to create fluency. Gary Stager observed:

> I wish I had 1 cent for every educator who has told me that her students "do a little Scratch." I always want to respond, "Call me when your students have done a lot of Scratch." The epistemological benefit of programming computers comes from long intense thinking. Fluency should be the goal.
> (Stager, 2017)

One of the paths to address these issues is the creation of partnerships between researchers and governments, since government officials need support in scaling efforts in order to go beyond oversimplified solutions. Guzdial (2017) is currently helping many states conduct landscape surveys[9] to determine the state of CEd in different parts of the country. He contends that policy decisions and coordination between different stakeholders would be much easier if landscape surveys were standard operating practice, as they allow states to gauge the growth of CEd offerings, PD programs, and enrollments. Yongpradit (2017) also noted that federal and state-level organizations urgently need technical assistance around creating certifications, growing the CEd teacher pipeline, and implementing curricula. Because CEd is such a new field, there are too few trained professionals and specialized organizations that can offer those services, leading to simplified and superficial implementations. But the issue of superficiality is also related to funding: Yongpradit expressed concern with current funding levels, noting that CEd requires more PD, standards development, and support for task forces to create implementation plans.

8 The paper examined data from a subset of about 5,000 users in January 2012 (Fields, Giant, & Kafai, 2013).

9 http://ecepalliance.org/resources/landscape-reports

3.4.4 Pluralism in CEd: Exploring New Domains and Tools

Another systemic issue raised by many interviewees and the literature was the importance of allowing for different ways of doing Computing, in terms of tools, programming languages, developmental levels, and approaches to organizing one's practice. In 1990, Sherry Turkle and Seymour Papert published an influential paper on *epistemological pluralism*, in which they described a study where children engaged in programming in a variety of non-canonical ways that were all ultimately successful (Turkle & Papert, 1990). Even though some children were violating traditional programming practice (the "bricoleurs"), they were doing so in a personally meaningful way that allowed them to create a strong connection with programming. Echoes of this influential paper were heard in almost all the interviews, and the principle of epistemological pluralism appears to have taken hold in CEd at the K–8 level. Grover (2017), however, pointed out that the epistemological pluralism approach needs to be combined with the teaching of some agreed-upon concepts and programming practices. When Resnick (2017) pointed out the need to keep pushing for epistemological pluralism, he noted that some systems only reward students for standard ways of doing coding (i.e., the smallest number of blocks when solving a puzzle), and some automated assessment programs still grade students solely based on the number and types of programming blocks they use. The interviewees also expressed the belief that traditional professional or college-level practices should not be automatically used in K–8 environments, since nontraditional approaches to programming (such as bricolage) may make sense only for younger students, even if advanced programmers might sometimes make use of these techniques as well (Berland, Martin, & Benton, 2013; Blikstein, 2011; Blikstein et al., 2014; Brennan, 2013; Graham, 2004).

3.4.5 Curriculum and Instructional Materials

The production of a quality curriculum and curricular materials is, for many interviewees, a key component for successful CEd implementations at scale. The interviewees noted that this is an area of significant and ongoing challenge despite efforts such as the K–12 CS Framework (K–12 Computer Science Framework Steering Committee, 2016):

> No one yet has written out a full, coherent K–12 curriculum built around a foundational framework. The K–12 CS Framework and the CSTA standards have laid out concepts, practices, and performance expectations but how do these things get manifested in curriculum and activities and experiences in K–12? That is a huge problem in computing right now that directly affects implementation.
> (Yongpradit, 2017)

Creating comprehensive curriculum materials is especially challenging because there is a natural tension between uniformity and the potential for customization to the learners' interests. Many interviewees noted the need to design culturally and personally relevant curricula that would cater to diverse populations (Buechley, 2017; Margolis, 2017; Resnick, 2017; Shapiro, 2017):

> The most important challenge is relating computing to [students'] culture and their identity. If you can get someone excited about something and engaged, they are incredibly motivated to learn.
> (Buechley, 2017)

Another important principle for curriculum design in CEd is the pedagogical approach in terms of how students will come into contact with the programming language. Pears et al.'s (2007) review of the literature found three major approaches: (a) focus on generic problem-solving; (b) focus on learning a particular programming language; and (c) focus on code production, or project-oriented CEd courses. As we discussed previously, the focus on higher-order problem-solving skills is problematic. Palumbo's (1990) review examined transfer between learning to program and problem-solving and concluded that more advanced forms of transfer (far or generalized transfer) should not be expected in introductory courses, since typically there is no time to develop such skills. In other words, if curricula aim for the transfer of problem-solving skills to other domains, explicit time and effort should be put into it. Scholarship has shown that positive results in problem-solving require a high involvement from teachers and well-developed theoretical foundations (Clements, 1990; De Corte & Verschaffel, 1989), as well as a considerable time investment. In one study, 150 hours of experience were needed to generate positive learning gains in problem-solving (Liu, 1997). Guzdial noted that this issue of programming and transfer is far from resolved, especially when the affinities and the unity of content and computation are not clear:

> Most people don't teach programming for transfer, and if they did, they would not be able to cover as much of programming. I think it is a zero-sum game: teach for programming fluency or teach for transferable problem-solving skills. You cannot get both in the same time.
> (Guzdial, 2017)

The second approach – focus on learning a particular programming language – is by far the most common. Textbooks, lesson plans, and assessments are designed based on the constructs of a programming language. This focus, common in introductory college courses and Advanced Placement (AP) classes in the USA, has been criticized by several interviewees as being too limited and too vocational. Buechley, for example, praised new initiatives (such as the new Advanced Placement Computer Science Principles course) that are moving AP classes away from the "one-language" model:

> So [Computer Science Principles] is a class that provides a different model of engaging with computing than the traditional computer science AP class did. And a model that is much more focused on foundational concepts and big ideas as opposed to the nuts and bolts of programming in a particular

> language. And because of that, it has the potential to provide more accessible pathways to more diverse kids, which is really important.
> (Buechley, 2017)

The third approach is code production or project-oriented learning. Instead of small assignments and tasks based on language constructs, or more general problem-solving training, students learn to create more complex systems to accomplish a task through projects. Even though this approach is harder to structure and assess, it seems to be more aligned with the approaches advocated by most interviewees. Resnick, for example, advocated for a project-oriented approach rather than small puzzles or language-based activities:

> There are a lot of schools where they do something with coding but it is done very superficially, just learning a few tricks of how to put some blocks together … but not really connecting in a deep way. [CEd should not be] just puzzles for kids learning to solve a problem, but a platform for expressing yourself.
> (Resnick, 2017)

Another common type implementation of code production or project-oriented approaches has been to make use of Computer Science-inspired mathematics and science practices. Science and mathematics as professional practices have been deeply transformed by computation, both in terms of the core disciplines themselves and the creation of entirely new fields such as bioinformatics, computational statistics, chemometrics, and neuroinformatics. Efforts to improve and modernize the teaching of science and mathematics should include computation as a core curricular component. Skills that can be developed through CS-infused science and mathematics include the ability to deal with open-ended problems, the creation of abstractions, recognizing and addressing ambiguity in algorithms, manipulating and analyzing data, and creating models and simulations (Weintrop et al., 2016).

Most of the interviewees identified infusing mathematics and science curricula with computation as a productive way to bring Computing to classrooms. diSessa (2018) highlighted that "there are people deeply enmeshed in non-CS disciplines, yet sufficiently expert with CS ideas and practices, to really get this agenda accomplished now." And Grover stated:

> It is very synergistic … computation makes the science and the math more real, authentic, and engaging. Students see aspects of the discipline that they would not see in the static form of learning from a textbook. Conversely, computation becomes alive because of the context in which it is used.
> (Grover, 2017)

Some interviewees expressed skepticism as to whether there are a sufficient number of available CEd teachers and whether it is possible to carve out space in the busy K–8 curriculum for a brand-new discipline. As a result, the interviewees noted that retraining science and mathematics teachers to add Computing to their teaching and generating new accompanying CS-infused lesson plans might be a more sustainable approach. Yongpradit (2017) also suggested that enabling teachers to receive dual certification in mathematics (or science) and Computing might be a positive alternative approach for addressing the current teacher shortage.

3.4.6 Teacher Development

One last (and crucial) systemic obstacle for large CEd implementations is the preparation of teachers. Ultimately, it is the interactions between teachers and students in classrooms that will determine whether students learn successfully. Thus, it is not surprising that the interviewees expressed the belief that teachers are the linchpin in any effort to implement or change CEd. The preparation, effective development, and retention of CEd teachers will need to be prioritized.

Teacher development was a central concern for most interviewees. Clancy (2017), Margolis (2017), and Yongpradit (2017) highlighted the challenges in building the CEd teacher workforce, and noted the need for teacher certification, training programs based on these certifications, and incentives for teachers to seek these qualifications. Guzdial (2017) highlighted the importance of preservice teacher development as the most viable way to achieve sustainability.

The need for equity in teacher development was also highlighted, since more affluent schools are more capable of offering high-quality programs. Interviewees noted that it is not enough to expose teachers to Computing content. Teachers need time to practice inclusive CEd, and these pedagogies should be interwoven into the entire teacher preparation program. Margolis (2017) also raised the need to educate teachers regarding bias, so that they can reflect on belief systems and perceptions about which students can excel in computing and how these beliefs would impact their relationships with students.

In general, there was concern about the rapid scaling of several initiatives and the capacity to prepare thousands of teachers adequately in a very short time. The interviewees argued that scaling too quickly disproportionately impacts underserved communities and populations that are historically excluded from STEM.

Margolis was particularly concerned with making equity a core tenet in teacher development, mentioning that, in her research, she encountered significant variability among teachers in their capacity for guiding deeper cognitive thinking. She found that teaching was particularly productive when teachers identified the specific Computing concepts for the students while they were learning them and discussed how they could relate the concepts to other areas of knowledge. The capacity to competently guide students in this way was found to be a predictor of student learning, but it varied considerably among teachers. Not surprisingly, teachers in less affluent areas were found to be the least prepared to enact these strategies in the classroom, in part because their districts had less funding for teacher PD. Margolis adds:

> Not only do teachers need to be introduced to the CS content, but they need to have time practicing pedagogies that are aimed at creating an inclusive

> CS learning environment, building on the assets, interests, and motivations of traditionally underrepresented students. Also, CS teacher PD must have equity and inclusion woven throughout everything that happens in PD, not just isolating this issue to a discrete one-hour discussion. For instance, as teachers are experiencing teaching lessons during PD, the other teachers who are in the roles as students or observers should be reflecting on their own experiences of inclusion (or not), thinking about their own students in their classrooms, and what works (or does not) to ignite the interest of all students. Also, teachers need time, and a safe learning environment, to reflect on all the biased belief systems associated with which students can and cannot excel in computing, to reflect on their own belief systems, and how belief systems impact their relationships with the students in their classrooms. Traditionally CS education has not been a place where these types of discussions or reflections have taken place, but they must if we are to broaden participation in computing.
> (Margolis, 2017)

Teacher development is further discussed in Chapter 25.

3.5 Conclusion

The year 2017 marked the 50th anniversary of the Logo programming language. In just five decades, an entirely new domain of knowledge has evolved from an idea in the minds of a few visionaries to national public policy. And while CEd is a relatively new discipline with a less substantial research base, there is much reason for optimism. Ensuring that we continue this progress, however, requires the commitment, work, and flexibility of a large number of stakeholders. We are now facing the growing pains intrinsic to progressing from pilot projects to large-scale implementations, and we must look and work beyond these growing pains to ensure that CEd fulfills its educational promise in sustainable and equitable ways.

Despite these challenges, CEd offers many advantages and the potential to transform learning environments and school work. Computing includes algorithms, design, data, making, creativity, and personal expression. CEd also facilitates productive collaboration in the classroom, connects to personally meaningful aspects of the lives of students, allows for new types of knowledge and assessments to be valued in schools, boosts the potential of project-based learning approaches, and opens possibilities of innovative ways to organize learning environments (e.g., Berland et al., 2013; Blikstein et al., 2014; Brennan, 2013; Buechley & Eisenberg, 2008; diSessa, 2000; Sherin, 2001; Turkle & Papert, 1990). Addressing and harnessing these advantages is important as our world becomes more technological and digital, and equitable participation requires computational fluency. This makes CEd necessary in K-8 not just as an elective subject, but as a mandatory topic. There is no question anymore about the importance of CEd and its place and need in public education, but there are differing opinions on why and how it should be done. Among the most prominent rationales for increasing access to CEd is that it can serve as a foundational literacy upon which other knowledge/activities can be built, and as a powerful context for profound, authentic, and interdisciplinary learning in other subjects. CEd can serve as an expressive, creative medium to allow young learners to express ideas in ways that are socially and culturally relevant, and it can also be a valuable tool for civic and political participation.

Given the importance of CEd, many of the interviewees believe that national rollouts of robust programs will require massive investment in the creation of state-level standards and curricula, teacher preparation and certification, software/hardware infrastructure, and research. It is not clear if all stakeholders are aware of the depth of the effort, but many feel that partial rollouts have the potential to increase social disparities and educational inequalities, privileging more affluent or well-resourced schools and districts. Additionally, although large-scale "CS exposure" programs are reaching millions of children, there is concern that they do not guarantee sustained engagement, particularly for underserved youth. Addressing these concerns requires better metrics, arms-length evaluation of programs, and more consensus on what constitutes success. In addition, exposure programs could benefit from follow-up activities, curricula, and sufficient resources to support deeper learning and stronger outcomes. And despite the growing demand for large-scale rollouts and the temptation of the adoption of one single implementation model, researchers advocate for a repertoire of well-studied and well-rationalized models that are sufficiently flexible to be adapted to multiple local contexts.

With an eye toward stronger outcomes, a reliance on high-quality curricula and assessments alone is not a guarantee of effective implementation. Education is always instantiated by teachers, so attention to pedagogy, teacher support, and the complex dynamics of adopting new curricula is crucial. Specifically, we found that teacher development is a key factor in the success of CEd, both preservice and in-service. In addition, the understanding of equity, inclusiveness, and unconscious biases about CS success are viewed as necessary to teacher development programs.

In sum, the time is ripe for thoughtfully targeted and comprehensive action to advance the CEd community. A large and diverse body of perspectives indicates that we must address the social, economic, and cultural barriers surrounding computing. If access and inclusiveness are addressed effectively, we can meet current and future workforce and citizenship demands. And we can do so in ways that equitably drive technological and social progress and give youth new avenues for personal expression and empowerment. Above all, we should avoid the trivialization and the oversimplification of computing knowledge, making it another missed opportunity to bring an exciting new set of practices, content, and cognitive tools to students.

This effort requires the cooperation and coordination of interdisciplinary, inter-sector teams that thoughtfully design, implement, evaluate, and learn from CEd initiatives. Only in this way can we achieve the hoped-for scale and sustainability and realize the ultimate vision of generations of researchers, practitioners, and policy-makers that have been trying, for the last 50 years, to bring computing to all students.

3.6 Acknowledgments

The authors wish to thank the 14 interviewees for their willingness to share a few centuries of accumulated knowledge about CEd, as well as their invaluable insight and vision for the future: Matthew Berland (University of Wisconsin – Madison), Leah Buechley (Rural Digital), Michael Clancy (University of California, Berkeley), Andrea “Andy” diSessa (University of California, Berkeley), Sally Fincher (University of Kent), Shuchi Grover (formerly SRI International), Mark Guzdial (Georgia Institute of Technology), Mike Horn (Northwestern University), Jane Margolis (University of California, Los Angeles), Mitchel Resnick (Massachusetts Institute of Technology), Sue Sentance (King’s College London), Ben Shapiro (University of Colorado, Boulder), David Weintrop (University of Maryland), and Pat Yongpradit (Code.org).

References

Berland, M. (2017). Phone interview with Paulo Blikstein.

Berland, M., Martin, T., & Benton, T. (2013). Using learning analytics to understand the learning pathways of novice programmers. *Journal of the Learning Sciences,* 22(4), 564–599.

Blikstein, P. (2011). Using learning analytics to assess students’ behavior in open-ended programming tasks. In *Proceedings of the 1st International Conference on Learning Analytics and Knowledge – LAK 2011* (pp. 110–116). New York: ACM.

Blikstein, P., Worsley, M., Piech, C., Sahami, M., Cooper, S., & Koller, D. (2014). Programming pluralism: Using learning analytics to detect patterns in novices’ learning of computer programming. *Journal of the Learning Sciences,* 23(4), 561–599.

Blikstein, P. (2013). Digital Fabrication and ’Making’ in Education: The Democratization of Invention. In J. Walter-Herrmann & C. Büching (Eds.). *FabLabs: Of Machines, Makers and Inventors* (pp. 203–221). Bielefeld: Transcript Publishers.

Blikstein, P. (2018). *Pre-College Computer Science Education: A Survey of the Field.* Mountain View, CA: Google LLC. Retrieved on 1 November 2018 from https://goo.gl/gmS1Vm

Brennan, K. (2013). Learning computing through creating and connecting. *Computer,* 46(9), 52–59.

Buechley, L. (2017). Phone interview with Paulo Blikstein.

Buechley, L., & Eisenberg, M. (2008). The LilyPad Arduino: Toward wearable engineering for everyone. *IEEE Pervasive Computing,* 7(2), 12–15.

Clancy, M. (2017). Phone interview with Paulo Blikstein.

Clements, D. H. (1990). Metacomponential development in a LOGO programming environment. *Journal of Educational Psychology,* 82(1), 141.

De Corte, E., & Verschaffel, L. (1989). Logo: A vehicle for thinking. In B. Greer & G. Mulhern (Eds.), *New Directions in Mathematics Education* (pp. 63–81). London/New York: Routledge.

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

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

Fincher, S. (2017). Phone interview with Paulo Blikstein.

Google LLC. & Gallup Inc. (2016). Diversity gaps in computer science: Exploring the underrepresentation of girls, Blacks and Hispanics. Retrieved from http://goo .gl/PG34aH

Graham, P. (2004). *Hackers & Painters: Big Ideas from the Computer Age*. Sebastopol, CA: O'Reilly Media.

Grover, S. (2017). Phone interview with Paulo Blikstein.

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

Guzdial, M. (2017). Phone interview with Paulo Blikstein.

Horn, M. (2017). Phone interview with Paulo Blikstein.

K–12 Computer Science Framework Steering Committee (2016). *K–12 Computer Science Framework* (978-1-4503-5278-9). Retrieved from http://k12cs.org/wp-content/ uploads/2016/09/K%E2%80%9312-Computer-Science-Framework.pdf

Liu, M. (1997). The effects of HyperCard programming on teacher education students' problem-solving ability and computer anxiety. *Journal of Research on Computing in Education,* 29(3), 248–262.

Maltese, A., & Tai, R. (2011). Pipeline persistence: Examining the association of educational experiences with earned degrees in STEM among US students. *Science Education,* 95(5), 877–907.

Margolis, J. (2017). Phone interview with Paulo Blikstein.

National Research Council (2006). *America's Lab Report: Investigations in High School Science*. Washington, DC: National Academies Press.

National Research Council (2012). *A Framework for K–12 Science Education: Practices, Crosscutting Concepts, and Core Ideas*. Washington, DC: National Academies Press.

Noonan, R. (2017). *STEM Jobs: 2017 Update (ESA Issue Brief # 02-17)*. Retrieved from www.esa.gov/reports/stem-jobs-2017-update

O'Neil, C. (2016). *Weapons of Math Destruction*. New York: Crown Publishing Group.

Palumbo, D. (1990). Programming language/problem-solving research: A review of relevant issues. *Review of Educational Research,* 60(1), 65–89.

Papert, S. (1980). *Mindstorms: Children, Computers and Powerful Ideas*. New York: Basic Books.

Pears, A., Seidman, S., Malmi, L., Mannila, L., Adams, E., Bennedsen, J., & Paterson, J. (2007). A survey of literature on the teaching of introductory programming. *ACM SIGCSE Bulletin*, 39(4), 204–223.

Resnick, M. (2017). Phone interview with Paulo Blikstein.

Seehorn, D., Carey, S., Fuschetto, B., Lee, I., Moix, D., O'Grady-Cunniff, D., …, Verno, A. (2011). *CSTA K–12 Computer Science Standards: Revised 2017.* Retrieved from www.csteachers.org/page/standards

Sentance, S. (2017). Phone interview with Paulo Blikstein.

Shapiro, B. (2017). Phone interview with Paulo Blikstein.

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

Stager, G. (2017). *A Modest Proposal*. Retrieved from http://stager.tv/blog/?p=4153

Turkle, S., & Papert, S. (1990). Epistemological pluralism: Styles and voices within the computer culture. *Signs: Journal of Women in Culture and Society,* 16(1), 128–157.

US Department of Labor (2007). The STEM workforce challenge: The role of the public workforce system in a national solution for a competitive science, technology, engineering, and mathematics (STEM) workforce. Retrieved from https://digitalcommons.ilr.cornell.edu/key_workplace/637/

Vogel, S., Santo, R., & Ching, D. (2017). Visions of computer science education: Unpacking arguments for and projected impacts of CS4All initiatives. In *Proceedings of the 48th ACM Technical Symposium on Computer Science Education – SIGCSE 2017* (pp. 609–614). New York: ACM.

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., & Wilensky, U. (2016). Defining computational thinking for mathematics and science classrooms. *Journal of Science Education and Technology,* 25(1), 127–147.

Wilensky, U. (1999, updated 2006, 2017). NetLogo [Computer software] (Version 6). Evanston, IL: Center for Connected Learning and Computer-Based Modeling. Retrieved from http://ccl.northwestern.edu/netlogo

Wilensky, U., & Papert, S. (2010). Restructurations: Reformulating knowledge disciplines through new representational forms. In *Proceedings of Constructionism 2010 Paris* (p. 15). Paris, France: American University of Paris.

Wing, J. M. (2006). Computational thinking. *Communications of the ACM,* 49(3), 33–35.

Yongpradit, P. (2017). Phone interview with Paulo Blikstein.