Computational thinking

Chris Proctor, Department of Learning and Instruction, University at Buffalo, SUNY, Buffalo, NY, United States

Introduction

As the individual and social importance of computing has grown since the end of the second world war, computer science emerged as an academic field in higher education and more recently in primary and secondary education as well. Modern educational institutions require operational definitions of content knowledge as the basis for curriculum standards, graduation requirements, standardized assessments, teacher certification standards, and other forms of setting goals and measuring progress. Computational thinking has emerged as *boundary object* (Star and Griesemer, 1989) between academic computer science and K12 education, providing a mutually-intelligible construct to practitioners with different needs, different disciplinary practices, and different contexts of use. Computer scientists have tended to approach computational thinking from the inside, introspectively articulating the kind of thinking made possible by thinking-with-computers, which is uniquely characteristic of the field of computer science. Educators and education researchers often approach computational thinking from the outside, asking how these new skills and knowledge should be defined and assessed, and how they might fit into the broader scope of K12 learning goals. The ongoing controversy over how to define computational thinking emerges partly from issues of translation across these contexts and perspectives, but it can also be understood as a proxy for a political debate about the role computers ought to play in society and what children ought to learn in school.

This article provides a history of computational thinking, a summary of computational thinking's role in educational policy and practice, role as a research construct, and concludes by considering the future of computational thinking. Throughout, computational thinking is considered in terms of two opposing questions:

1. How does computational thinking define the heart of computer science?
2. How does computational thinking describe the borderlands?

History of computational thinking

The precursors to computational thinking predate the creation of functional computers. Ada Lovelace, credited with writing the first computer program in 1843, recognized that planning processes of symbolic manipulations (algorithms) represented a form of human thought distinct from the machinery which would carry them out. The machine, she wrote, "can do whatever we know how to order it to perform" (1843, p. 722), and imagined a future in which computers would be able to process symbolic representations of mathematics, language, and music. Although no computer capable of running symbolic programs would be built during her lifetime, Lovelace imagined a future in which new forms of thinking would be brought about by "more intimate practical acquaintance with the powers of the engine" (as cited in Fuegi and Francis (2003), p. 24). Lovelace's famous correspondence with Charles Babbage, who was more concerned with the mechanical details of building a computer, foreshadows the later distinction between computational thinking as a set of abstract knowledge, skills, strategies, and dispositions, and programming as an activity which concretely puts computational thinking to use.

The successful military use of computers in World War II led to speculation about how computers might also transform civilian life. Vannevar Bush, who led the United States's wartime research and development, anticipated that computers would help people find, remember, and organize the exponentially-growing amounts of information generated by society (1945). Perlis argued in the early 1960s that programming ought to be a standard part of a liberal education, enabling everyone to frame problems in computational terms (Guzdial, 2008). Engelbart's (1962) goal of using computers to augment human intellect led to the invention of personal computing and ultimately today's consumer computer ecosystem. As computers became smaller and less expensive, educator-technologists such as Papert (1980) and Turkle (1984) explored how computers might offer powerful learning experiences to children.

Papert, the first to use the phrase "computational thinking" (1980, p. 182), was interested in using computers not "to put children through their paces" (1980, p. 19) but to serve as partners in thought providing contexts and representations through which children might form personal relationships with powerful ideas. He was dismayed at the tendency of school curricula to focus on the delivery of denatured and decontextualized "content," similar to Freire's (1968) critique of the "banking concept of education." Writing about probability, for example, Papert argued that "one of the most powerful ideas in our intellectual heritage is (not untypically) disempowered in its school presentation, where it is reduced to shallow manipulations that seldom connect to anything the student experiences as important" (1996, p. 4). Papert argued that computers could serve as "micro-worlds" in which children might learn powerful ideas as easily and naturally as children learn French by growing up in France. Building off the work of Piaget and Vygotsky, Papert and later Constructionists hypothesized that computer cultures might produce new developmental trajectories for children and new futures for society.

The current era of computational thinking begins in 2006, when Wing's argument for computational thinking as a "universally applicable attitude and skill set everyone, not just computer scientists, would be eager to learn and use" (p. 33) catalyzed legislatures, policymakers, researchers, and businesses around the goal of *CS for All*, or adding computer science to the fundamental competencies every child should develop throughout primary and secondary education. Wing defines computational thinking as involving "solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science" (p. 33). The article offers a long list of examples of computational thinking, starting with concepts such as recursion, parallel processing, type checking, and indirect addressing, which have specific technical meanings in computer science but which could be applied more generally if one frames a problem in computational terms. She encourages such abstraction, providing several examples of how concepts from computer science can be used to reinterpret situations from everyday life and envisioning a future in which computational ways of thinking about the world have become widespread and privileged: "Computational thinking will have become ingrained in everyone's lives … when trees are drawn upside down," suggesting that tree-as-data-structure will have eclipsed the ecological concept of a tree (p. 34).

Wing offers a number of criteria for locating the boundaries of computational thinking which have been broadly influential in shaping the development of the concept in policy and research, as well as presciently indexing some of its key definitional tensions. First, she argues that computational thinking is "conceptualizing, not programming" (p. 35), an idea later interpreted by numerous standards documents to mean that computational thinking and programming are disjoint—a starker distinction than Wing may have intended. Furthermore, computational thinking is "fundamental, not a rote skill" (p. 35), anticipating the argument that computational thinking ought to be the emphasis in general and interdisciplinary education, leaving programming to the subset of students preparing for specialized careers in computer science and engineering. Wing suggests that computational thinking "complements and combines mathematical and engineering thinking" (p. 35), an assertion explored in subsequent literature on what is distinct about computational thinking and how it relates to other forms of disciplinary thinking. Finally, Wing argues that computational thinking is about "ideas, not artifacts" (p. 35), echoing prior arguments (e.g., Dijkstra, 1989) that computer programs ought to be taught as formal mathematical systems and not in terms of their behaviors.

Wing's definition of computational thinking would form the basis of a rapid adoption of computational thinking as a primary and secondary learning goal around the world. As the concept was put into practice in educational policy and research, tensions and contradictions emerged around how computational thinking should be defined and how it should be taught. Computational thinking found its way into schools through standalone computer science courses as well as through interdisciplinary implementations across the curriculum, leaving researchers to reconcile narrow and broad conceptualizations.

Computational thinking in educational policy and practice

The goal of universal K12 computer science education was taken up by a coalition of public- and private-sector advocates of computing education such as the Association for Computing Machinery, Code.org, the Computer Science Teachers association (computer scienceTA), the International Society for Technology in Education (ISTE), the UK's Computing at School, and the Australian Curriculum, Assessment, and Reporting Authority. These groups pursued a strategy of uniting behind a clear, uniform definition of the essential skills and knowledge of computer science, and they selected computational thinking as this definition despite unresolved details about the meaning and scope of the term, noted above (Barr and Stephenson, 2011). Computer science was introduced as a standalone course (particularly at the high school level) as well as an interdisciplinary literacy which could be taught across the curriculum; as a result, computational thinking was tasked with articulating both the essential heart of computer science as well as its interdisciplinary potential.

Major reports commissioned in the United States (Wilson et al., 2010) and the United Kingdom (The Royal Society, 2012) found that schools were far from preparing students with the computing skills they would need in the anticipated digital economy, and called on governments to create K12 computer science standards organized around computational thinking. Following initial standards developed by computer scienceTA (2011) and ISTE (2011) identifying computational thinking as a core strand of computer science, a broad coalition of stakeholders developed the K12 Computer Science Framework (2016) to guide individual states in defining their computer science content standards. This framework proposed seven core practices at the heart of computer science, and identified four of them as comprising computational thinking: recognizing and defining computational problems; developing and using abstractions; creating computational artifacts; and testing and refining computational artifacts (p. 68). The framework largely followed Wing's earlier definition of computational thinking, emphasizing that computational thinking is distinct from computer science, that computer science is the best context for learning computational thinking, and that computational thinking should be included in standards for other subjects as well.

These efforts to define computational thinking laid the groundwork for computational thinking educational policy initiatives around the world. A 2019 review of global computational thinking educational policy initiatives (Hsu et al., 2019) found that five countries had implemented national computational thinking curricula: the United Kingdom, Australia, the United Arab Emirates, Finland, and Russia. Others, including India, Chile, Japan, China, Korea, Taiwan, and Hong Kong, have processes underway (Hsu et al., 2019; So et al., 2020). Hsu et al. (2019) found wide conceptual variety in how computational thinking is defined, as well as how computational thinking is understood to relate to computer science and to programming. National curricular reforms vary in the extent to which they implement computational thinking by adding computer science as a new subject area or embed interdisciplinary computational thinking across existing subject areas.

In addition to interdisciplinary applications, computational thinking has been used as a bridge between formal academic computer science and other contexts. Although many authors caution that the mere use of a computer (in or out of school) does not imply the presence of computational thinking, informal contexts such as clubs and online communities have been particularly important places for learners who have been marginalized by formal computer science to develop interest and experience which may serve as the foundations of computational thinking learning (National Academies of Sciences, Engineering, and Medicine, 2021). Informal learning environments have also been particularly welcoming to sociocultural reframings of computational thinking (Kafai and Burke, 2013; Repenning et al., 2010), discussed in more detail below.

A lack of access to early computing experiences also plays a role in inequitable computer science participation (Margolis and Fisher, 2003), leading to increased emphasis on computational thinking learning opportunities earlier in high school (e.g., introductory courses such as Exploring Computer Science (Goode and Margolis, 2011)), the middle and primary grades, and even early childhood education. Wang and Proctor's (in press) review of computational thinking in early childhood education finds significant tensions between standard framings of computational thinking which prioritize interpreting phenomena in terms of computational models and the holistic developmental priorities of early childhood education. Instead, approaches grounded in Papert's understanding of computational thinking as a relationship between learners and computers has found more traction (Bers, 2017).

Assessment of computational thinking

Implementation of computational thinking standards and curricula is still quite recent so standardized assessments of computational thinking are generally not yet in place. As of 2010, "assessments for computer science education [were] virtually nonexistent" (Wilson et al., 2010, p. 14). Assessment tends to drive implementation in K12 education (Eisenhart et al., 1996; Grover and Pea, 2013), so it is not surprising that the development of computational thinking assessments has been a policy priority as well as an active area of research. The development of assessments has also helped surface tensions within definitions of computational thinking, as an assessment requires precisely operationalizing a construct and resolving ambiguities in standards.

Brennan and Resnick's (2012) exploration of how computational thinking might be assessed remains by far the most cited framework for approaching the assessment of computational thinking. Brennan and Resnick identify three dimensions of computational thinking–concepts, practices, and perspectives—and consider several process-based strategies for assessing them, including studying students' design processes, artifact-based interviews, and design scenario tasks. Brennan and Resnick's focus on the process of artifact creation rather than the product is motivated by their primary interest in supporting youth in creating personally-meaningful artifacts, but their argument aligns with other arguments for attention to process. An artifact created with code does not constitute evidence of computational thinking without also considering the process with which it was made ("K-12 Computer Science Framework," 2016; Salac and Franklin, 2020). Similarly, Denning (2017) argues for understanding computational thinking as a competency characterized by how people think about problems (specifically, employing abstractions which represent a problem in terms of a computational model), not just what they know or what they have produced. This view presents a challenge to the decontextualized, content-based standardized testing characteristic of national policy initiatives.

Another difficulty in assessing computational thinking is that computational thinking appears to be tightly-bound to the modality in which computational thinking has been learned (Weintrop and Wilensky, 2015), reinforcing the importance of the relationship between cognition and the medium of representation (Wilensky, 2010). Despite the argument for the broad applicability of computational thinking, students' skills do not transfer easily between programming languages or representations, nor is there evidence that computational thinking improves intelligence in general (Guzdial, 2015; Pea and Kurland, 1984). To address this dependence on modality, Werner et al. (2012) developed a performance assessment embedded in the programming environment and task structure students had been using to learn computational thinking.

Several recent reviews conclude that substantial difficulties remain in the assessment of computational thinking. In terms of Brennan and Resnick's framework of concepts, practices, and perspectives, the vast majority of computational thinking assessments around the world focus on concepts, with little consensus on which concepts are included and how they are defined (Cutumisu et al., 2019). Most assessments are designed to test content knowledge without attending to process, and very few have been validated (Lye and Koh, 2014; McGill et al., 2019). Computational thinking assessment has made substantial recent progress in early childhood education, with efforts to stabilize the construct (Bers, 2017) and a validated assessment (Relkin et al., 2020).

Computational thinking as a research construct

As suggested by the uneven state of computational thinking assessment, the uptake of computational thinking in educational policy has outpaced its development as a research construct (Wilson and Guzdial, 2010). Since Wing's 2006 article, researchers have engaged in a spirited debate about what computational thinking ought to mean and how it ought to be defined, measured, and taught. However, based on the distribution of citations, this debate has had more urgency within the computer science education community than in the broader literature on education (Saqr et al., 2021). Empirical research on computational thinking often begins by recounting the history of these disagreements before proceeding with an ad-hoc measure of computational thinking. The two questions organizing this article, *How does computational thinking define the knowledge, skills, and practices at the heart of computer science?* And *How does computational thinking describe the borderlands?*, represent proximal and distal poles in a fundamental question underlying much of the controversy around computational thinking: how broad is computational thinking? (Proctor and Blikstein, 2018; Voogt et al., 2015).

How does computational thinking define the heart of computer science?

Between roughly 2006–2013, the academic conversation around computational thinking focused largely on how the construct articulated the heart of computer science, itself a dynamic and loosely-bounded discipline. In 2009 the US National Research Council convened a workshop aimed at defining the scope and nature of computational thinking. While the participants did not reach consensus on either topic, they did largely agree on the centrality of computational thinking in computer science, and that computational thinking is fundamentally a process of abstraction. In a subsequent article, Wing (2011) developed this idea into a concise definition of computational thinking:

> Computational thinking is the thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can effectively be carried out by an information-processing agent (p. 1).

The workshop also considered the relationship between computational thinking and programming. Although examples of unplugged pedagogies (by which computational thinking is taught without the use of computers (Bell et al., 2009)) were shared, the workshop appeared to generally agree that while computational thinking is distinct from programming, code plays an important role in computational thinking. This is embedded in Wing (2011) definition, as programming is a primary (though not the only) mode of representing solutions which can be interpreted by a computer. In the words of Mitchel Resnick, "computational thinking is more than programming, but only in the same way that language literacy is more than writing … programming, like writing, is a means of expression and an entry point for developing new ways of thinking" (p. 13).

These two findings—that computational thinking describes a thought process of applying abstractions to problems, and that programming plays an important role in computational thinking—are notable in retrospect for the extent to which they have been minimized in the implementation of computational thinking in policy and practice. In the decade following Wing's 2006 article, numerous frameworks proposed taxonomies of what should be included within computational thinking, with substantial variation (Angeli et al., 2016; Barr and Stephenson, 2011; Brennan and Resnick, 2012; Grover and Pea, 2013; Kalelioglu et al., 2016; Repenning et al., 2016; Seiter and Foreman, 2013; Selby, 2012). Abstraction (the "undisputed" keystone of computational thinking (Grover and Pea, 2013, p. 39)) was always included as a core element of computational thinking; algorithms, data, decomposition, parallelization, testing and debugging, and control structures were often included as well (Zhang and Nouri, 2019). A recent review proposed six facets of computational thinking: decomposition, abstraction, algorithm design, debugging, iteration, and generalization (Shute et al., 2017). Nevertheless, K12 computational thinking curricula and assessments disproportionately address computational thinking concepts, suggesting that computational thinking is taught as a body of knowledge rather than a process (Lye and Koh, 2014).

Recent implementations of computational thinking have also de-emphasized programming for principled and for practical reasons. In contrast to Resnick's quotation above suggesting programming is like writing, Lu and Fletcher (2009) view programming as a specialized practice, unnecessary and uninviting for beginners: "programming is to Computer Science what proof construction is to mathematics, and what literary analysis is to English" (p. 260). Others have suggested that teaching programming focuses students' attention on low-level details which are becoming less important as interaction with computers progresses to higher-level abstractions (Pears et al., 2021). In primary and secondary education, teaching computational thinking without programming has had an easier time finding purchase in interdisciplinary contexts and as the basis for pedagogical approaches which are manageable for teachers without a background in computer science.

As computational thinking gained widespread adoption in K12 education globally, concerns emerged from within the academic computer science community that the desire for broad applicability had led to an overly-vague definition of computational thinking which had lost its focus on the essential process of framing problems in computational terms (Tedre and Denning, 2022). Denning (2017) envisions computational thinking defined in terms of competencies–skills and sensibilities demonstrated in practice—while noting that assessments of computational thinking are largely focused on static content knowledge. He takes issue with curricula based on these assessments structured as "progressions of increasingly sophisticated learning objectives" rather than sequences of skill acquisition (p. 36). Even if programming is not necessary in computational thinking, Aho (2012) emphasizes the centrality of computational models (formal systems which can run automatically) in computational thinking, arguing that interdisciplinary computational thinking requires a discipline's constructs discipline to be represented in terms of a computational model. Computational models are widely-accepted in adjacent STEM fields such as the life sciences, but in the social sciences and particularly the humanities, whose questions more often address interpretation, positionality, and subjectivity, the role of computational models is less clear.

Grover and Pea's (2013) review of computational thinking in *Educational Researcher* marks a milestone in the research trajectory of computational thinking, taking stock of progress and projecting the research agenda ahead, while also mediating the perspectives of the computer science education community and the broader community of educational research. After summarizing the prior research on computational thinking (much of which, they note, was focused on defining computational thinking), Grover and Pea proposed broadening the scope of computational thinking research by focusing on interest, identity, and stereotypes contributing to a lack of diversity in computational thinking enrollment, computational thinking research grounded in sociocultural and situated theories of learning, and interdisciplinary computational thinking. At the same time, they called for more empirical research on K12 computational thinking practice, including the development of evidence-based curricula, pedagogies, and tools for K12 computer science and models for integrating computational thinking into professional development and teacher preparation. While Grover and Pea's review does not demarcate a clean division, it marked a shift in emphasis from computational thinking as it relates to computer science to computational thinking as it relates to K12 education more broadly.

How does computational thinking describe the borderlands?

The term *borderlands*, referencing Anzaldúa (1987), points to liminal spaces and the possibilities of hybridity, crossing, or syncretism drawing on multiple sources. The borderlands of computational thinking are at the edge of what counts from the perspective of computer science, sometimes crossing into other disciplines or communities of practice with their own perspectives and priorities. The borderlands include teachers and students in schools and informal contexts (who may be more interested in creating meaningful experiences than fidelity of implementation), those oppressed by race, gender, or other categories (who may be indifferent or hostile to their powerful computing practices being identified as computational thinking), computational thinking in interdisciplinary contexts (where computational thinking may play an instrumental role in the service). As computational thinking has made its way into K12 practice over the last decade, the community of researchers interested in computational thinking has expanded to include those whose primary concern is these spaces and peoples.

Following the 2009 National Research Council workshop on the scope and nature of computational thinking, another workshop was convened to consider the pedagogical aspects of computational thinking (National Research Council, 2011). Although K12 teachers were not represented, the workshop affirmed the importance of teachers and pedagogical skill, and recognized the potential for computational thinking in formal primary and secondary education, as well as informal spaces. As the visibility of computer science practitioners grew within the computing education research world (led by the Computer Science Teachers Association), an alternative emerged to the theoretical debate over how to define computational thinking. The Use-Modify-Create framework (Lee et al., 2011), in which a group of researchers based in K12 schools and informal contexts chose to focus not on what computational thinking means in theory, but what it looks like in practice. The report aimed to help educators understand and teach computational thinking in the context of children's socioeconomic and cultural contexts and their developmental trajectories. In the framework, students first *use* computational artifacts created by others in order to understand their external functionality, then *modify* artifacts to understand their inner workings, and finally *create* their own artifacts. This strategy grounds computational thinking in a sociocultural account of learning, aiming to support students' inbound trajectories of participation and developing a sense of authorship and agency with the artifacts and the code in which they are implemented. The epistemic importance of teachers has only grown, with recent calls to substitute the top-down, "abstract and agenda-driven" processes for defining computational thinking and designing pedagogical approach with emergent definitions grounded in local needs and realities (Wilkerson et al., 2020, p. 265).

A substantial body of literature has documented the persistent inequities present in K12 computer science education. Equity-oriented researchers studying how mainstream computer science learning environments tend to support dominant-culture youth at the expense of their marginalized peers have found it useful to frame learning in terms of sociocultural constructs such as identity-building (Shaw and Kafai, 2020; Vakil, 2020) and participation in communities of practice (Scott and White, 2013), and to examine the relationship between computing and structures such as race and gender (Benjamin, 2019; Margolis and Fisher, 2003). However, these sociocultural and critical perspectives are substantially less present in the literature on computational thinking. The K12 Computer Science Framework (2016) recognizes the importance of practices such as collaboration, communicating, and fostering an inclusive computing culture, but clearly demarcates them from computational thinking. Similarly, the AP Computer Science Principles course, which was designed around computational thinking, views creativity and collaboration as "pedagogical strategies to be used to develop a diverse, appealing, and inclusive classroom environment" rather than as part of computational thinking (College Board, 2020). In empirical research, race, gender, and other categories are more often treated as contextual factors or moderating variables, rather than as formations shaping the nature of activity.

If computational thinking is essentially about the process of formulating problems so that their solutions can be carried out by an information-processing agent, sociocultural and critical constructs are relevant to who is formulating which problems, and to what end. Vakil's (2018) agenda for a justice-centered approach to equity in computer science education would emphasize the social and political implications of computational thinking, and the disciplinary, civic, and political identities which authorize students and legitimize problems as suitable for computational thinking. Similarly, ethnocomputing applies the processes of computational thinking to the "knowledges, practices, and designs of Indigenous or vernacular cultural systems" (Lachney et al., 2022, p. 112). Several alternatives to computational thinking have also been proposed, such as *computational participation* (Kafai and Burke, 2013) *computational action* (Tissenbaum et al., 2019), and *computational literacies* (Kafai and Proctor, 2021) which view constructs such as identity, interest, power, and formations (Omi and Winant, 2001) such as race and gender as integral to the practices of computational thinking.

Finally, interdisciplinary borderlands of computational thinking have been receiving increasing attention at the K12 level, following the increasing importance of computing in research across disciplines (National Research Council, 2010). Computational thinking has been deployed in an effort to "highlight the integral relationship between computing and other fields" (Wilkerson et al., 2020, p. 264). K12 STEM disciplines have eagerly adopted computational thinking. For example, the Next Generation Science Standards include computational thinking among the core practices of science (Weintrop et al., 2016). Domain-specific "teaspoon languages" have recently been developed which "add a teaspoon of computing to other subjects." The goal is that by the time that US students take a computer science class (typically, in high school or undergraduate), they will have had many programming experiences, have seen a variety of types of programming languages, and have a sense that "programming isn't hard" (Guzdial and Shreiner, 2022).

Three framings of computational thinking

The two questions organizing this article are positioned respectively within computer science and within K12 education. They represent different stakeholders in the research community, and are oriented toward different future research trajectories, raising the prospect of fragmentation of the computing education research community. In response to a proposal that computing education ought to avoid *theory bias* by limiting its engagement with educational theory (Nelson and Ko, 2018), Kafai et al. (2020) argued that all accounts of learning are grounded in theoretical frames and instead proposed a commitment to *theory dialog*. Their goal was to avoid the so-called "literacy wars" and "math wars" in which researchers committed to different paradigms were unable to find common ground with respect to disciplinary learning goals, what would constitute evidence of success, or sometimes even legitimate methods of research.

Toward this end, Kafai, Proctor, and Lui identified three framings of computational thinking: cognitive, situated, and critical, which distinguish the theoretical underpinnings of various projects within the computer science education research community. Cognitive computational thinking seeks to provide students with an understanding of key computational concepts and competencies. Situated computational thinking understands learning in terms of creative expression, interest development, and identity authorship and participation. Critical computational thinking recognizes that computing is not an unequivocal social good, and proposes an analytical approach to the values, practices, and infrastructure underlying computation as part of a broader goal of education for justice and participation in shaping our shared future.

Kafai, Proctor, and Lui represented these framings as three concentric circles, suggesting units of analysis at the scale of the individual, the collaborative group, and the city or society. These three framings are loosely parallel to the concepts, practices, and perspectives proposed by Brennan and Resnick (2012). Just as the vast majority of computational thinking assessments currently focus exclusively on concepts, cognitive framings of computational thinking remain dominant. As the social importance of computing continues to grow and educational stakeholders from outside of computing education participate in the academic conversation around computational thinking, the authors hope that situated and critical research will be legible to researchers whose work is largely cognitive and dialog across framings will be possible.

The first decades of the 21st century have seen widespread adoption of computational thinking as an educational priority around the world, accompanied by a research base whose growth continues to accelerate (Saqr et al., 2021). As the social importance of computing continues to grow, computational thinking appears to be well on its way toward becoming a fundamental literacy at the heart of K12 education. With this integration the two questions anchoring this article, asking how computational thinking defines the heart of computer science and how computational thinking describes the borderlands, are likely to be increasingly in dialog with each other.

References

Aho, A.V., 2012. Computation and computational thinking. Comput. J. 55, 832–835. https://doi.org/10.1093/comjnl/bxs074.

Angeli, C., Voogt, J., Fluck, A., Webb, M., Cox, M., Malyn-Smith, J., Zagami, J., 2016. A K-6 computational thinking curriculum framework: implications for teacher knowledge. J. Educ. Technol. Soc. 19, 12.

Anzaldúa, G., 1987. Borderlands: La frontera. Aunt Lute, San Francisco.

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, 48–54.

Bell, T., Alexander, J., Freeman, I., Grimley, M., 2009. Computer Science Unplugged: School Students Doing Real Computing Without Computers, vol. 10.

Benjamin, R., 2019. Race After Technology: Abolitionist Tools for the New Jim Code. Social Forces.

Bers, M.U., 2017. Coding as a Playground: Programming and Computational Thinking in the Early Childhood Classroom. Routledge.

Brennan, K., Resnick, M., 2012. New frameworks for studying and assessing the development of computational thinking. In: Proceedings of the 2012 Annual Meeting of the American Educational Research Association. Vancouver, Canada, p. 25.

Bush, V., 1945. As we may think. In: The Atlantic Monthly, vol. 176, pp. 101–108.

College Board, 2020. AP Computer Science Principles Course and Exam Description.

Cutumisu, M., Adams, C., Lu, C., 2019. A scoping review of empirical research on recent computational thinking assessments. J. Sci. Educ. Technol. 28, 651–676. https://doi.org/10.1007/s10956-019-09799-3.

Denning, P.J., 2017. Remaining trouble spots with computational thinking. Commun. ACM 60, 33–39. https://doi.org/10.1145/2998438.

Dijkstra, E.W., 1989. On the cruelty of really teaching computing science. Commun. ACM 32, 1398–1404.

Eisenhart, M., Finkel, E., Marion, S.F., 1996. Creating the conditions for scientific literacy: a re-examination. Am. Educ. Res. J. 33, 261–295.

Engelbart, D.C., 1962. Augmenting Human Intellect: A Conceptual Framework, pp. 64–90.

Freire, P., 1968. Pedagogy of the Oppressed.

Fuegi, J., Francis, J., 2003. Lovelace & Babbage and the creation of the 1843 "Notes". IEEE Ann. Hist. Comput. 16–26.

Goode, J., Margolis, J., 2011. Exploring computer science: a case study of school reform. J. Educ. Resour. Comput. 11, 1–16. https://doi.org/10.1145/1993069.1993076.

Grover, S., Pea, R., 2013. Computational thinking in K–12: a review of the state of the field. Educ. Res. 42, 38–43. https://doi.org/10.3102/0013189X12463051.

Guzdial, M., Shreiner, T., 2022. Integrating computing through task-specific programming for disciplinary relevance: considerations and examples. In: Yadav, A., Berthelsen, U.D. (Eds.), Computational Thinking in Education: A Pedagogical Perspective, pp. 172–190.

Guzdial, M., 2008. Education paving the way for computational thinking. Commun. ACM 51, 25. https://doi.org/10.1145/1378704.1378713.

Guzdial, M., 2015. Learner-centered design of computing education: research on computing for everyone. Synth. Lect. Hum. Centered Inf. 8, 1–165. https://doi.org/10.2200/S00684ED1V01Y201511HCI033.

Hsu, Y.-C., Irie, N.R., Ching, Y.-H., 2019. Computational Thinking Educational Policy Initiatives (CTEPI) Across the Globe. TechTrends. https://doi.org/10.1007/s11528-019-00384-4.

ISTE, 2011. ISTE Standards for Computer Science Educators.

K-12 Computer Science Framework, 2016.

Kafai, Y.B., Burke, Q., 2013. The social turn in K-12 programming: moving from computational thinking to computational participation. In: Proceeding of the 44th ACM Technical Symposium on Computer Science Education—SIGCSE '13. ACM Press, Denver, Colorado, USA, p. 603. https://doi.org/10.1145/2445196.2445373.

Kafai, Y.B., Proctor, C., 2021. A revaluation of computational thinking in K-12 education: moving towards computational literacies. Educ. Res. https://doi.org/10.3102/0013189X211057904.

Kafai, Y.B., Proctor, C., Lui, D., 2020. From theory bias to theory dialogue: embracing cognitive, situated, and critical framings of computational thinking in K-12 CS education. ACM Inroads 11, 44–53.

Kalelioglu, F., Gülbahar, Y., Kukul, V., 2016. A framework for computational thinking based on a systematic research review. Baltic J. Modern Comput. 4, 14.

Lachney, M., Green, B., Allen, M.C., Foy, L., 2022. Ethnocomputing and computational thinking. In: Yadav, A., Berthelsen, U.D. (Eds.), Computational Thinking: A Pedagogical Perspective. Routledge, pp. 112–135.

Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., Malyn-Smith, J., Werner, L., 2011. Computational thinking for youth in practice. ACM Inroads 2, 32–37. https://doi.org/10.1145/1929887.1929902.

Lovelace, A., 1843. Notes on I. Menabrea's "Sketch of the analytical engine invented by Charles Babbage, Esq". Taylor Sci. Memoirs 3, 1843.

Lu, J.J., Fletcher, G.H., 2009. Thinking about computational thinking. In: Proceedings of the 40th ACM Technical Symposium on Computer Science Education, pp. 260–264.

Lye, S.Y., Koh, J.H.L., 2014. Review on teaching and learning of computational thinking through programming: what is next for K-12? Comput. Hum. Behav. 41, 51–61. https://doi.org/10.1016/j.chb.2014.09.012.

Margolis, J., Fisher, A., 2003. Unlocking the Clubhouse: Women in Computing. MIT Press.

McGill, M.M., Decker, A., McKlin, T., Haynie, K., 2019. A gap analysis of noncognitive constructs in evaluation instruments designed for computing education. In: Proceedings of the 50th ACM Technical Symposium on Computer Science Education—SIGCSE '19. ACM Press, Minneapolis, MN, USA, pp. 706–712. https://doi.org/10.1145/3287324.3287362.

National Academies of Sciences, Engineering, and Medicine, 2021. Cultivating Interest and Competencies in Computing: Authentic Experiences and Design Factors. National Academies Press, Washington, D.C. https://doi.org/10.17226/25912.

National Research Council, 2010. Report of a Workshop on the Scope and Nature of Computational Thinking. National Academies Press.

National Research Council, 2011. Report of a Workshop on the Pedagogical Aspects of Computational Thinking. National Academies Press.

Nelson, G.L., Ko, A., 2018. On use of theory in computing education research. In: Proceedings of the 2018 ACM Conference on International Computing Education Research—ICER '18. ACM Press, Espoo, Finland, pp. 31–39. https://doi.org/10.1145/3230977.3230992.

Omi, M., Winant, H., 2001. Racial formation. In: Essed, P., Goldberg, D.T. (Eds.), Race Critical Theories: Text and Context. Blackwell, Malden, MA, pp. 123–145.

Papert, S., 1980. Mindstorms: Children, Computers, and Powerful Ideas. Basic Books, Inc.

Papert, S., 1996. An exploration in the space of mathematics educations. Int. J. Comput. Math. Learn. 1. https://doi.org/10.1007/BF00191473.

Pea, R.D., Kurland, D.M., 1984. On the cognitive effects of learning computer programming. New Ideas Psychol. 2, 137–168. https://doi.org/10.1016/0732-118X(84)90018-7.

Pears, A., Tedre, M., Valtonen, T., Vartiainen, H., 2021. What Makes Computational Thinking So Troublesome? https://doi.org/10.13140/RG.2.2.20480.35842.

Proctor, C., Blikstein, P., 2018. How broad is computational thinking? A longitudinal study of practices shaping learning in computer science. In: Kay, J., Luckin, R. (Eds.), Rethinking Learning in the Digital Age. Making the Learning Sciences Count. International Society of the Learning Sciences, London, UK, pp. 544–551.

Relkin, E., de Ruiter, L., Bers, M.U., 2020. TechCheck: development and validation of an unplugged assessment of computational thinking in early childhood education. J. Sci. Educ. Technol. 29, 482–498. https://doi.org/10.1007/s10956-020-09831-x.

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, SIGCSE '10. Association for Computing Machinery, New York, NY, USA, pp. 265–269. https://doi.org/10.1145/1734263.1734357.

Repenning, A., Basawapatna, A., Escherle, N., 2016. Computational thinking tools. In: 2016 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC). IEEE, Cambridge, United Kingdom, pp. 218–222. https://doi.org/10.1109/VLHCC.2016.7739688.

Salac, J., Franklin, D., 2020. If they build it, will they understand it? Exploring the relationship between student code and performance. In: Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education. ACM, Trondheim Norway, pp. 473–479. https://doi.org/10.1145/3341525.3387379.

Saqr, M., Ng, K., Oyelere, S., Tedre, M., 2021. People, ideas, milestones: a scientometric study of computational thinking. J. Educ. Resour. Comput. 21, 1–17. https://doi.org/10.1145/3445984.

Scott, K.A., White, M.A., 2013. COMPUGIRLS' standpoint: culturally responsive computing and its effect on girls of color. Urban Educ. 48, 657–681. https://doi.org/10.1177/0042085913491219.

Seiter, L., Foreman, B., 2013. Modeling the Learning Progressions of Computational Thinking of Primary Grade Students. ACM Press, p. 59. https://doi.org/10.1145/2493394.2493403.

Selby, C.C., 2012. Promoting computational thinking with programming. In: Proceedings of the 7th Workshop in Primary and Secondary Computing Education on—WiPSCE '12. ACM Press, Hamburg, Germany, p. 74. https://doi.org/10.1145/2481449.2481466.

Shaw, M., Kafai, Y., 2020. Charting the identity turn in K-12 computer science education: developing more inclusive learning pathways for identity. In: International Conference of the Learning Sciences. Nashville, US (Conference Cancelled).

Shute, V.J., Sun, C., Asbell-Clarke, J., 2017. Demystifying computational thinking. Educ. Res. Rev. 22, 142–158. https://doi.org/10.1016/j.edurev.2017.09.003.

So, H.-J., Jong, M.S.-Y., Liu, C.-C., 2020. Computational thinking education in the Asian Pacific region. Asia Pac. Educ. Res. 29, 1–8. https://doi.org/10.1007/s40299-019-00494-w.

Star, S.L., Griesemer, J.R., 1989. Institutional ecology, translations' and boundary objects: Amateurs and professionals in Berkeley's museum of Vertebrate Zoology, 1907–39. Soc. Stud. Sci. 19, 387–420.

Tedre, M., Denning, P., 2022. Computational thinking: a professional and historical perspective. In: Yadav, A., Berthelsen, U.D. (Eds.), Computational Thinking: A Pedagogical Perspective. Routledge, pp. 1–17.

The Royal Society, 2012. Shut Down or Restart? The Way Forward for Computing in UK Schools. The Royal Society.

Tissenbaum, M., Sheldon, J., Abelson, H., 2019. From computational thinking to computational action. Commun. ACM 62, 34–36. https://doi.org/10.1145/3265747.

Turkle, S., 1984. The Second Self: Computers and the Human Spirit, 20th anniversary ed., first MIT Press ed. MIT Press, Cambridge, Mass.

Vakil, S., 2018. Ethics, identity, and political vision: toward a justice-centered approach to equity in computer science education. Harv. Educ. Rev. 88, 26–52. https://doi.org/10.17763/1943-5045-88.1.26.

Vakil, S., 2020. "I've always been scared that someday I'm going to sell out": exploring the relationship between political identity and learning in computer science education. Cognit. InStruct. 38, 87–115. https://doi.org/10.1080/07370008.2020.1730374.

Voogt, J., Fisser, P., Good, J., Mishra, P., Yadav, A., 2015. Computational thinking in compulsory education: towards an agenda for research and practice. Educ. Inf. Technol. 20, 715–728. https://doi.org/10.1007/s10639-015-9412-6.

Wang, X.C., Proctor, C., (in press). Computational thinking (CT) meets young children: critical review of conceptualization and implementation of CT in early childhood. Early Child. Res. Q.

Weintrop, D., Wilensky, U., 2015. Using commutative assessments to compare conceptual understanding in blocks-based and text-based programs. In: Proceedings of the Eleventh Annual International Conference on International Computing Education Research, ICER '15. ACM, New York, NY, USA, pp. 101–110. https://doi.org/10.1145/2787622.2787721.

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., Wilensky, U., 2016. Defining computational thinking for mathematics and science classrooms. J. Sci. Educ. Technol. 25, 127–147. https://doi.org/10.1007/s10956-015-9581-5.

Werner, L., Denner, J., Campe, S., Kawamoto, D.C., 2012. The fairy performance assessment: measuring computational thinking in middle school. In: Proceedings of the 43rd ACM Technical Symposium on Computer Science Education. ACM, pp. 215–220.

Wilensky, U., 2010. Restructurations: Reformulating Knowledge Disciplines Through New Representational Forms, vol. 15.

Wilkerson, M.H., D'Angelo, C.M., Litts, B.K., 2020. Stories from the field: locating and cultivating computational thinking in spaces of learning. Interact. Learn. Environ. 28, 264–271. https://doi.org/10.1080/10494820.2020.1711326.

Wilson, C., Guzdial, M., 2010. How to make progress in computing education. Commun. ACM 53, 35–37. https://doi.org/10.1145/1735223.1735235.

Wilson, C., Sudol, L.A., Stephenson, C., Stehlik, M., 2010. Running on Empty: The Failure to Teach K–12 Computer Science in the Digital Age. ACM, New York, NY, USA. https://doi.org/10.1145/3414583.

Wing, J., 2011. Research notebook: computational thinking—what and why. Link Mag. 6, 20–23.

Zhang, L., Nouri, J., 2019. A systematic review of learning computational thinking through scratch in K-9. Comput. Educ. 141, 103607. https://doi.org/10.1016/j.compedu.2019.103607.