From Theory Bias to Theory Dialogue: Embracing Cognitive, Situated, and Critical Framings of Computational Thinking in K-12 CS Education

Yasmin Kafai, *University of Pennsylvania*, Chris Proctor, *Stanford University*, and Debora Lui, *University of Pennsylvania*

The increased interest in promoting CS education for all has been coalescing around the idea of "computational thinking." Several framings for promoting computational thinking in K-12 education have been proposed by practitioners and researchers that each place different emphases on either (1) skill and competence building, (2) creative expression and participation, or (3) social justice and ethics. We review each framing and how the framings structure the theory space of computational thinking. We then discuss how CS education can leverage the explanatory potential that each framing offers to the implementation and evaluation of learning, teaching, and tools in computing education. Our goal is to help CS education researchers, teachers, and designers unpack and leverage the complexities of this theory space (rather than ignoring it) while also addressing broader educational concerns regarding diversity, providing new directions for how students and teachers can actively participate in designing their digital futures, and directing current computing education efforts towards a more humanistic orientation.

INTRODUCTION

Promoting computer science education has become a global initiative with the goal to make it a 21st century literacy. Under the umbrella of "computational thinking" [75], initiatives around the world propose that every child should learn a core set of computational skills and use them across the curriculum as well as in everyday life. Alongside the active debate over how (or whether) to define computational thinking, hundreds of studies (e.g. [32, 59]) have investigated applications of computational thinking for K-12 CS education. Within these efforts, different theoretical perspectives have become more visible, so much that Nelson and Ko [45:31] have argued: "while theory can accelerate our fields's progress and increase its rigor, if not used carefully, it can also inhibit progress in subtle but important ways." More specifically, Nelson and Ko [45] argue that over-reliance on educational theory within computer science education research can inhibit the progress within the field by dividing researchers' attention between contributing to general learning theory versus developing new designs, overshadowing domain-specific educational knowledge, and introducing "theory bias" in peer review.

While Nelson and Ko [45] highlight important concerns, we take issue with the notion that any interpretation of learning and thinking can be meaningful without considering its theoretical underpinnings. Learning of any subject matter is framed through various theoretical lenses—or metaphors as Sfard [61] argued—each of which carry tacit assumptions and beliefs not only regarding how people learn best, but also for why and for what purposes. In turn, these theoretical perspectives consequently guide any interpretation or understanding of instructional activities, actions and tool designs. Rather than ignoring this intricate connection between theory and design, we therefore advocate foregrounding the diversity of theoretical perspectives in learning and teaching that exist within K-12 CS education and research. Following Haraway [26], our goal here is not to push any kind of 'objective' truth about the best practices in CS Education, but rather to acknowledge that all perspectives and approaches within the field are partial and contingent. An intervention whose results appear lackluster from one theoretical perspective might be profoundly impactful from another. Only by recognizing these partial perspectives can we truly reach critical transformational opportunities for K-12 CS education and research.

Attention to educational theory is particularly important as momentum is growing behind Wing's [75:33] argument: "to reading, writing, and arithmetic, we should add computational thinking to every child's analytical ability." If computational thinking is to become a new literacy [31, 36] added to the canon of textual, mathematics and science literacies, we need to frame computational thinking beyond an understanding of computational concepts and practices needed for digital content production, to include an understanding of the values, biases, and histories embedded in the digital technologies. Becoming literate is not just about the pragmatics of reading and writing text (or code) but also about how these skills are inherently contextualized within particular personal and political dimensions [58]. From this vantage point, if computational literacy is only configured as instrumental [69], it will miss critical aspects that have emerged including inequities caused or exacerbated by the societal impact of computing (e.g.,[41]). While much attention has been paid to the lack of diversity in practice and participation in K-12 CS education, the diversity of theoretical framings or lenses through which we design, examine, and evaluate computing education has received far less scrutiny.

In this paper, we use the concept of computational thinking to illustrate how theoretical framings direct our attention to different, but equally important aspects in learning and teaching within K-12 CS education. Theoretical framing is needed to articulate educational goals, and therefore to evaluate the quality of pedagogical designs. We disagree with Nelson and Ko [45] that there can be a theory-free evaluation of learning, or that some designs can be objectively better than others, outside of any theoretical framing. As a first step, we identify and describe three prevalent framings of computational thinking that we have found within the larger landscape of CS education: (1) Cognitive computational thinking seeks to provide students with an understanding of key computational concepts, practices, and perspectives thereby emphasizing skill building and competencies which will be useful in college and future careers; (2) Situated computational thinking stresses personal creative expression and social engagement as a pathway in becoming computationally fluent building on youth interest in digital media and production; and (3) 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.

We illustrate each framing with examples from various studies and discuss how these framings of computational thinking have functioned as design heuristics that provide specific directives for curricular initiatives that inform the design of learning and teaching tools, materials and activities. We then consider how these framings are an integral part of the larger theory space of efforts promoting K-12 computational thinking and how they should be considered in dialogue with one another rather than in opposition. Based on this understanding, we offer suggestions for how to proceed forward with a more holistic view of not only what computational thinking should be, but also directions for it might be studied or taught moving into the future.

THREE FRAMINGS OF COMPUTATIONAL THINKING

Over a decade ago, Wing [75:33] proposed the term computational thinking as "involving solving problems, designing systems and understanding human behavior that draws on concepts fundamental to computing" (p. 33) to the CS community. Wing is certainly not the first person to describe a skill set needed to design and implement computations—which has also been referred to as procedural or algorithmic thinking [2]. Efforts to define computer science's unique ways of thinking and practicing are part of a "long quest" within the discipline, aiming to distinguish itself from engineering and mathematics and developing its independent identity [64]. While some see computational thinking as specific to the discipline [13], others such as Wing assign it more general purpose status that is not necessarily tied to machines [11].

Settling these differences is beyond the scope of this paper, but what is relevant to our work is how computational thinking has been taken up within K-12 education. Papert [47:182] is credited with introducing the term wanting "to integrate computational thinking into everyday life." Papert [47] and others envisioned early on that computational ideas could serve as a tool for not only learning mathematics [16] but also a wide range of other subjects in new ways [1, 14, 62]. This general purpose application of computational thinking garnered much traction in bringing the first wave of computers into schools in the 1980's but also generated considerable critique because of its lack of empirical evidence for transfer [49].

In many of today's national initiatives, standards, curricula, and courses, computational thinking has again been adopted as a general purpose skill which forms the basis for competence building that all students need to learn [70]. This approach often ignores other framings which provide different directions for designing and understanding learning and teaching. Drawing on prominent educational learning theories, we have broadly categorized these other framings as situated and critical computational thinking. In the following sections, our goal is to outline each framing of computational thinking and then to provide examples of how each framing has been employed in current research studies. We describe how research involving the same programming tool and context, Scratch [53], asks different questions, seeks different learning outcomes, and results in different curricular activities when computational thinking is framed in different ways. Scratch is a block-based programming language and community, which has attracted over ten millions of kids in the creation of personalized projects, including animations and video games, that are shared online. These distinct framings of CT shape not only how activities using Scratch are designed, but also what roles learners play, and what is valued in terms of learning outcomes.

COGNITIVE COMPUTATIONAL THINKING

The dominant framing of computational thinking, reflecting the majority of research in CS education [65], is cognitive. The cognitive framing of computational thinking seeks to provide students with an understanding of key computational concepts, practices, and perspectives [5] thereby emphasizing skill building and competencies which will be useful in college and future careers. This direction draws from cognitive research traditions that already dominated efforts to introduce programming in the 1980's (e.g. [63]). Here computational thinking is seen as a form of problem solving performed by individual students [25]. Instructional activities are developed to introduce students to computational concepts like loops, recursion, conditionals, and data structures, and practices such as iteration, abstraction and automation, and responsible interactions. A large number of related efforts also promote the integration of computational thinking into STEM disciplines [71]. Many national standards and curricula such as Code.org's CS Discoveries [9] have adopted this direction and mapped out learning progressions and pathways for how students should develop computational thinking, starting as early as kindergarten.

Most studies focused on student learning within Scratch have highlighted this cognitive emphasis, looking at students' understanding of foundational CS concepts. A large part of this research has focused on assessment and evaluation of students' programming ability and comprehension of basic and advanced coding constructs (e.g., variables, conditional logic), through activities such as think-aloud interviews, creating functional open-ended projects, and engaging with design scenarios [5, 24]. Others have stressed how the particular contexts of Scratch learning could promote cognitive gains, for instance, looking at how block or text-based programming languages can support learning of CS concepts in different ways [72, 73], or how the specifics of interface and game design on Scratch can be used to motivate and assess learning of computational thinking concepts [74]. Although the cognitive framing of computational thinking typically assesses learning on an individual basis, it would be unfair to suggest that cognitive approaches necessarily imply that instruction is isolated and decontextualized. On the contrary, most of these studies involve students creating individual projects and artifacts rather than learning these skills within artificially isolated contexts. However, the goal of all of this activity is to increase individual comprehension of CS concepts and competent programming performance, something that distinguishes it from the next framing described below.

SITUATED COMPUTATIONAL THINKING

An alternative proposition to the cognitive emphasis has been a situated framing of computational thinking, which sees value in students developing computational fluency through designing and programming shareable digital artifacts. This framing draws from constructionist [47] and connected learning theories [29], which emphasize interest-driven and peer-supported activities. Here computational thinking is seen as a vehicle for personal expression and connecting with others alongside and intersecting a plurality of other literacy practices [22]. Learning key computational concepts and practices are often situated within the design of digital applications shared with authentic audiences in person or over social networks. For these reasons, this approach has also been called 'computational participation' [8] in order to emphasize the social purpose of computational designs and interactions in which learners engage. Efforts in this direction emerged largely from promoting CS education outside of school in community technology centers and online communities, and from a recognition that inequitable access to opportunities to participate, develop interest, and have one's identities supported are a root cause of the lack of diversity in computer science [41, 42]. Example curricula include the *Creative Computing Guide* [4] which situates students' introduction to computational thinking through a variety of game design and storytelling activities or *Stitching the Loop* [35] activities that engage students in crafting and coding personalized electronic textiles.

This situated approach has become yet another popular area of research within Scratch, which has emphasized the particular socio-cultural contexts in which this activity occurs, thereby emphasizing personal meaning and creative expression. Within some studies, this focus is accomplished by explicitly pushing the link between creativity and computing and working to broaden perspectives on the field of Computer Science at-large. For instance, Giannakos, Jaccheri, and Proto [21] developed a Scratch activity where children worked alongside adults to create interactive artworks for the purposeful goal of encouraging youth to become digital creators through programming. Related efforts have included situating Scratch within the context of students' other interests, including music [17, 31], storytelling [4, 76], or, most often, video games. These approaches have resulted in tens of thousands of digital artifacts created by children in the Scratch online community, referencing popular commercial game franchises and narratives [36].

Emphasis on situated computational thinking in Scratch is additionally accomplished through highlighting the social interactions therein, whether structuring forms of online collaboration and feedback through the site itself [21], sharing and creative remixing of other people's projects [8], or creating games and tangible controllers that are explicitly meant to engage an in-person audience [17]. While students' comprehension of CS concepts and abilities to program are important here, the main goal of these efforts is to emphasize how computing is a tool that can be used to express students' interests and identities to others within their communities. Notably, allowing personalized pathways into computing is meant to engage individuals who might otherwise be excluded from the field and explicitly done in an attempt to promote equity within the field [60].

CRITICAL COMPUTATIONAL THINKING

Finally, critical computational thinking has emerged more recently as another framing which places students' computational thinking in the traditions of critical pedagogy, which emphasize both an examination of and resistance to oppressive power structures [20] and production-oriented media literacy, which highlights how youth agency can be acquired through the process of creating and disseminating media content [6, 44]. Here, computational thinking is seen as a potential channel for engaging with the political, moral and ethical challenges of the world whether food insecurity or gentrification through the production of digital, multimedia products. In line with Paris's [48] argument that truly supporting marginalized students requires helping them to understand and contest the forces which marginalize them, some researchers have argued that situated computational thinking does not go far enough in confronting forces such as racism and sexism [68]. Activities that have adopted this approach include an after-school project in which youth interviewed residents and worked together with designers and programmers to visualize gentrification in their neighborhood [39], and a mobile app that would collate available out-of-school programs and opportunities for youth living in under-resourced communities [67]. Curricula such as Exploring Computer Science [46] have additionally addressed some of these issues by designing socially relevant and meaningful computation activities for marginalized students.

In terms of Scratch, there have been far fewer efforts to promote critical approaches to computational thinking, something which is also true within the wider field of CT research. One notable effort in this direction are the aforementioned work of Lee and Soep [39:480], which used Scratch in their efforts to push youth to "conceptualize, create, and disseminate digital projects that break silences, expose important truths, and challenge unjust systems." The goals here were to design activities that supported students' agency in developing their own computational artifacts in order to address personally relevant social justice issues, for instance, games that highlight issues such as racial profiling and undocumented laborers [43]. Importantly, the primary emphasis here was developing critical content using computation that responds to structural issues in the world (the 'what'), rather pushing students to analyze and understand the actual underlying infrastructure that supports everyday computation (the 'how').

More recently however, there have been calls to expand critical computational thinking such that it does focus on this second goal, namely, 'pulling back the curtain' of the technological mechanisms underlies our existing computational systems in order to understand how these may cause inequities in and of itself [68]. This includes, for instance, considering current event issues like how implicit bias might be embedded in crime-prediction software that police use, or how mass surveillance by social media can create openings for election hacking [68]. We can see, therefore, how earlier work in critical CT considered critical content creation as the primary goal, with the "skill building in coding and design" [39:480] as a desirable but secondary outcome, while the latter emphasis highlights a cognitive understanding of underlying concepts of CT and its uses in the world as key to becoming a more critical practitioner of computation.

UNDERSTANDING THE THEORY SPACE OF COMPUTATIONAL THINKING

Each framing offers valuable, but different insights into what learning and teaching computational thinking can and should be about. One striking commonality is that the learning of computational thinking within each of these three framings is often situated in the context of designing applications such as instructional software or games [33, 37] rather than learning code for its own sake. This contextualization is a stark departure from how computational thinking was taught during the first wave of computer science in schools in the 1980s. At that time, if students engaged with computing at all, it was in the context of writing short programs in which they learned computational concepts and practices, disconnected from the rest of the curriculum, their personal media interests, or any social relevance [55].

Where differences emerge between these three framings is how they balance their goal of promoting basic programming competence and understanding (something that is necessary for all three frameworks), with understanding how these skills can be used both for personal/social enrichment and to address issues within the world at-large. For instance, the emphasis in cognitive framings is on individual competency with computational skills and knowledge; building personal relationships with ideas is framed as part of the design leading to learning, rather than the learning itself. Situated framings center the construction of long-lasting and meaningful relationships with CS—a critical feature for a STEM field that historically has been an exclusive clubhouse [10]. But fostering personal connections alone is no guarantee for inclusion as we know from studies of online creative learning communities such as Scratch where content is often lacking cultural relevance [40]. However, we do not argue that the expanding focus from cognitive to situated to critical is simply progress toward better framings. We know that participation alone will not guarantee that novice programmers have access to key computational concepts [19] or pathways into more advanced forms of computational participation [18]. Likewise, designing social justice-focused applications [43] takes advantage of the benefits of critical media production but does not always guarantee more in-depth computational understanding (one exception is [12]).

While our discussion of different framings suggests equal relevance, within CS education the framings have not received equal attention: the cognitive framing by far outpaces other theoretical approaches in published CS education research [65, 68]. One possible reason for this dominance of cognitive framings in CS education is that when the first wave of research started in the 1980s, cognitive theory had just gained traction for gaining better insights into students' thinking and problem solving across different academic disciplines [27]. CS education researchers followed suit, most likely finding resonance with the cognitive perspectives featuring the individual mind as an information processing unit, not unlike a computer itself. However, critics of cognitive educational research have highlighted some of its weaknesses, namely that learning is not just an individual enterprise but situated in social interactions and contexts [41, 42]. This critique gave rise to new emphases within educational research, namely a socio-cultural perspective which recognizes the need for authentic learning practices—an aspect also highly valued by all—and the realization that learning is about becoming a member of a community of practice with shared goals and values.

Sfard [61] most clearly articulated these distinctions between cognitive and situated framings of learning as two metaphors of "acquisition" and "participation", respectively, in educational learning research. She pointed out how cognitive approaches treat knowledge as a property that learners acquire since it focuses on individuals, while situated approaches to learning see participation as a key process in which knowledge is negotiated between members of a community since it focuses on social interactions. While not originally included in Sfard's [61] analysis, critical approaches to learning might add "action" to "acquisition" and "participation," emphasizing that what is learned and how it is learned and valued reflects the particular norms, values, and power structures of a society. When a society is unjust, education ought to be oriented toward understanding and challenging injustice. We sought to make visible the different epistemological commitments of each learning perspective and how it related to the framings of computational thinking, an overview of which is provided in Table 1.

Table 1: Overview of Learning Perspectives in Framing Computational Thinking

Frame	Unit of Analysis	Epistemology	Priorities	Computational Thinking
Cognitive	Individual learners	Skills, competencies, knowledge of a particular discipline	Measurable, transferrable skills, economic opportunity	Computational concepts (algorithms, abstraction) and practices (remixing, iteration)
Situated	Communities of practice, activity systems, learning ecologies	Practices, participation, preparation for future learning	Equity, interest, identity development, creativity	Creating personally-meaningful applications, building communities, supporting social interactions, play
Critical	Society at-large: existing structures of power, privilege, and opportunity (race, gender, social class, ability)	Awareness of ideologies, strategies for social action	Justice, critical understanding, enacting social change	Understanding and critique of existing computational infrastructures, creating applications to promote thriving, awareness, and activism

Furthermore, rather than pitting the different metaphors and framings against each other, we follow Sfard's [61:12] conclusion that we should embrace, and not ignore, other theoretical perspectives in education research:

> As researchers we seemed to be doomed to living in a reality of variety of metaphors. We have to accept the fact that the metaphors we use while theorizing may be good enough to fit small areas, but none of them suffice to cover the entire field. In other words, we must satisfy ourselves with only local sensemaking. A realistic thinker knows he or she has to give up the hope that the little patches of coherence will eventually combine into a consistent global theory. It seems the sooner we accept the thought that our work is bound to produce a patchwork of metaphors rather than a unified, homogenous theory of learning, the better for us and for those whose lives are likely to be affected by our work.

By putting these views into dialogue with one another, we can acknowledge that each offers a partial perspective that can answer different questions about learning and teaching that can lead us to a more full and complete picture of how we can succeed together in this space. Our proposal in moving forward is to engage in the building of a "patchwork of metaphors" as Sfard [61] suggested by putting the various framings into a more inclusive dialogue rather than in exclusive opposition to each other. In the following section, we discuss three examples—designing tools, learning at scale, and teaching computing—to illustrate what such a dialogue could concretely look like.

PUTTING THE FRAMINGS IN DIALOGUE

We focus on how framings in conversation with one another can create opportunities for us to examine computational thinking from more than one perspective, ultimately building a more sophisticated foundation for CS education research and practice. Our first example illustrates different interpretations in understanding the same programming tool for promoting critical data literacy, or analytical approaches to understanding how digital data is leveraged and used online. In this study, researchers developed a new set of "community blocks" in Scratch which explicitly makes transparent the collection, calculation, and dissemination of participation data common in many massive online communities that users could use while programming their personal projects [12]. As the researchers discovered, users of these blocks became more aware of numerous issues surrounding how digital data is both gathered and used by systems such as Scratch, including issues surrounding privacy and data sharing and possible avenues for exclusion generated through certain data-driven algorithms [28]. Within this study, we can see how the different framings can contribute not only individually but also collectively to a more sophisticated understanding of how computational tools can assist in promoting computational thinking on multiple levels.

In this scenario, the use of the blocks supported students' *cognitive* understanding since integration of the community blocks with regular programming blocks to require a solid understanding of computational concepts and programming practices. Even when the pedagogical goals are cognitive, situated approaches may be more effective [51]. The community blocks also supported situated use of Scratch, since users created customized projects with the blocks, whether an ice cream visualization or dress-up game project that used social metrics to determine the number of scoops or the project viewer's purchasing power [12]. Finally, these blocks were essential in promoting *critical* engagement with CT, since they got users to further consider the larger computing infrastructures in which they engage every day, for instance, questioning the intentions of Scratch designers and other users in designing some community features (e.g., friends, favorites), as well as the affordances and constraints of different types of technology in controlling social interactions online [28]. The design of community blocks in Scratch therefore served as a tool that could promote goals within all three framings, something which might serve a model for future design and analysis work.

Our second example leverages the different scales of analysis promoted within each learning perspective and framing. Cognitive framings of computational thinking can create opportunities for us to think about what is happening at the level of an individual learner, focusing specifically on their mental constructs and understandings. In contrast, situated computational thinking can allow us to step back and consider what is occurring amongst multiple people, whether within physical spaces like students in a classroom making games for one another, or within digital space. Finally, critical computational thinking with its focus on societal structures can allow us to zoom further out, considering how these individuals and groups are situated within larger structures of capital, resources, and ideologies spanning neighborhoods, nations, and globalized networks. While one can apply these different framings with other kind of scales—for example, considering the cognition of all elementary school students in the United States, or how an individual becomes marginalized by her online social networks on the basis of her race or ethnicity—access to these multiple framings of CT can give us new tools to not only ask but also answer these questions.

Consider an imaginary teen who is a regular participant in library-based technology-focused afterschool program. A cognitive framing would allow us to zero in on the mental processes of this teen, looking to see how her actual activities (e.g., remixing existing racing games) can support learning particular CS concepts. A situated framing would allow us to step back to consider how this teen's work on Scratch can help her create connections online (e.g., the siblings from Thailand whose games she's remixed) and in-person (e.g., showing off her game to her younger brother and his friends). Finally, a critical framing would allow us to ask how this particular afterschool program is situated within the larger network of CS opportunities for this teen (e.g., if this is the first time she has ever been given the opportunity to code, and if this motivates her to recruit other people from her community into the program). By considering these different framings in dialogue then, we can look at the computing life of this teen from multiple scales, highlighting a more holistic perspective on her computational thinking engagements (see Figure 1).

Finally, considering the three framings in dialogue with one another allows CS education researchers and practitioners to take into account multiple ways of teaching computational thinking. While cognitive research on computational thinking has often considered a 'best practices' approach toward teaching CS concepts and practices, considering all three perspectives can open up pedagogies of computational thinking oriented to different but equally valid epistemologies. For instance, introducing particular CS concepts such as loops, variables, and Boolean logic might benefit from highly scaffolded debugging activities, while promoting critical analyses of online surveillance and privacy concerns might require another approach of asking students to research their own personal engagements with social media. Additionally, considering these framings in tandem might also help us all to consider approaches toward computing that have not yet been explored and developed.

Early work mostly took place within informal learning environments and emphasized creative expression with e-textiles, thereby giving students free rein to explore existing projects and objects, and incorporate their own interests into designing a personalized artifact [11]. In critical examinations of e-textiles learning, we focused on their gendered nature [25] and relation to traditional crafting practices [23]. Moving into the more formal environment of computer science classrooms required a cognitive lens, looking at students' understanding of foundational computing concepts and practices [7, 14] and developing a series of increasingly complex projects [5, 11]. The examination of teaching practices adopted a more situated framing of how teachers supported peer learning and promoted personal expression when teaching CS concepts in instructor consultations and prescribed milestones [5].

These examples illustrate how adopting a 'patchwork of metaphors' [61] could be useful for CS researchers, designers, and educators. This is particularly important as CS moves into the K-12 space, which is populated a diverse patchwork of priorities and stakeholders. K-12 CS education advocates invoke a variety of rationales for its importance [3], and recent efforts to design and implement K-12 CS suggest that implementations which fail to deeply engage with this plurality of perspectives may encounter indifference or opposition [50]. We do not advocate that researchers abandon their epistemological commitments but rather suggest that they maintain their distinct theoretical framings while also considering others. Having multiple framings would allow their work to be understandable by the broader community, making it easier to affirm areas of agreement, and question assumptions in a mutually-comprehensible way. This would go a long way in addressing central concerns about potential theory bias voiced by Nelson and Ko [54]. Instead of trying to limit ourselves to one-size-fits-all approach that aims

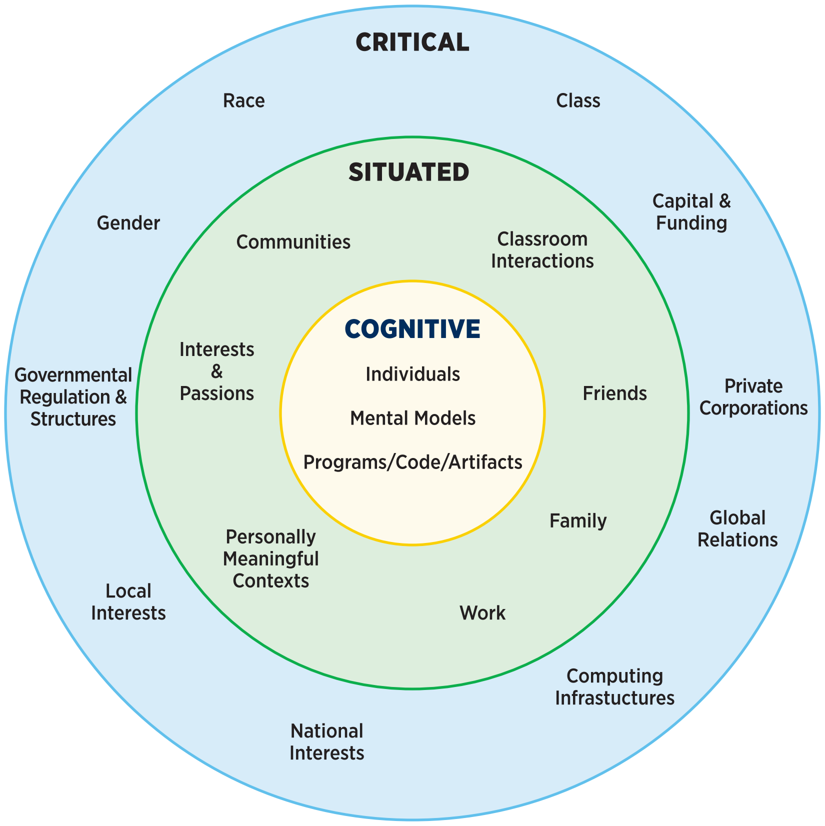

Figure 1: Framings of computational thinking represent different scales of analysis

to fulfill the computational thinking needs of all students, we can focus on which questions are important and which theories and methods are best suited to address them.

CONCLUSIONS

Nelson and Ko's [54] concerns that overemphasis on theory building may hinder innovation, understanding, and evaluation are important and need to be addressed within the CS education research community. However, we take issue with the implicit suggestion that design could be pursued separately from theoretical engagement. Sandoval's [57] conjecture mapping distinguishes between design conjectures (hypotheses about how a design might result in mediating processes) and theoretical conjectures (hypotheses about how mediating processes might result in learning outcomes). In concrete terms, we can collect analytics about how students use a programming interface to answer a design conjecture, but a theoretical conjecture (supported by further evidence) is required to support the claim that a particular kind of learning outcome has occurred. Design-based research, as practiced in the learning sciences, "often aims to innovate not just processes of instruction but the kinds of outcomes desired from instruction" [57:24].

Furthermore, we argue that foregrounding theoretical perspectives are what make the priorities and perspectives of diverse stakeholders visible within the context of research and policy. The diversity of perspectives on computational thinking is a resource, and not a stumbling block, for understanding the complex ecologies in which learning and teaching in CS education is situated. Placing an individual research project's framing within a broader theory space, could make it possible to engage with legitimate and important questions based in other framings.

In this paper, we unpacked different framings of computational thinking that have been in use in the CS education community. Rather than seeking conceptual unity in computational thinking, we highlighted the different ontological commitments that cognitive, situated and critical framings bring to computational thinking and illustrated how these contextualize research with programming tools, design of applications, and classroom implementations. Our approach to theory dialogue is in line with recent efforts which have proposed expanded framings of computational thinking such as computational participation [34], computational making [56], or computational action [66] and begin to cross the boundaries established around each framing. For instance, computational action [66] seeks to connect cognitive and critical aspects while computational participation [34] connected situated and critical aspects.

We proposed that research on computational thinking should make room for multiple framings and privilege interdisciplinary perspectives. Ultimately, we believe our proposal of theory dialogue is a matter of getting CS education researchers to understand the different scales and perspectives in which they are working when studying computational thinking. Recent calls for expanding computational thinking into computational literacy [15, 30, 38, 52, 69] potentially offer a construct which could accommodate the three framings discussed in this paper. For example, diSessa's [14] analysis of literacy focuses on "material intelligence," or thinking with a representational medium. His discussion of cognitive and social aspects of material intelligence could easily be expanded to align with the framings of computational thinking presented here.

However, if the CS education research community is to profit from this shift, literacy ought to be used as the basis for dialogue, not internecine battles. Scribner [58], writing in the context of the so-called "literacy wars" between advocates of phonics and contextualized whole-language instruction, chose to discuss literacy in terms of metaphors instead of definitions. Like Sfard [61], Scribner argued that "conflicts and contradictions are intrinsic to...an essentialist approach" [58:7]. Ultimately, the tensions giving rise to these definitional questions indicate the growing societal importance of computing and the maturation of the field of CS education research. "Points of view about literacy as a social good, as well as a social fact, form the ground of the definitional enterprise. We may lack consensus on how best to define literacy because we have differing views about literacy's social purposes and values" [58:8]. Advancing computational literacy situates the learning and teaching of computer science in broader, inherently-contested questions about the role of education in a democratic society. ❖

Acknowledgements
The writing of this paper was supported by grant #1509245 from the National Science Foundation to Yasmin Kafai. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the National Science Foundation, the University of Pennsylvania, or Stanford University.

References

1. Abelson, H. and diSessa, A. *Turtle geometry: The computer as a medium for exploring mathematics*. (Cambridge: MIT Press, 1986).
2. Aho, A. Ubiquity symposium: Computation and computational thinking. in *Ubiquity 2011*. (2011).
3. Blikstein, P. *Pre-College Computer Science Education: A Survey of the Field*. (2018).
4. Brennan K., C Balch, C., and Chung, M. *Creative Computing 3.0*. (Cambridge: Harvard University Press, 2019).
5. Brennan, K. and 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, 2012), 25.
6. Buckingham, D. *Media education: Literacy, learning and contemporary culture*. (Cambridge: Polity Press, 2003).
7. Flórez,F., Casallas, R., Hernández, M., Reyes, A., Restrepo, S., and Danies, G. Changing a Generation's Way of Thinking: Teaching Computational Thinking Through Programming. *Review of Educational Research* 87, 4 (2017-08), 834–860. doi: 10.3102/ 0034654317710096.
8. Burke, Q. and Kafai, Y. The writers' workshop for youth programmers: digital storytelling with scratch in middle school classrooms. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education (SIGCSE '12)*. (New York: ACM Press, 2012), 433–438. doi: 10.1145/2157136.2157264.
9. Code.org. *CS Discoveries*. https://code.org/educate/csd. Accessed: 2019 June 1.
10. National Research Council. *Report of a workshop on the scope and nature of computational thinking.* (Washington: National Academies Press, 2010).
11. Curzon, P., Bell, T., Waite, J., and Dorling, M. Computational Thinking. in *Cambridge Handbook of Computer Science Education Research*, edited by S. Fincher and A. Robbins (Cambridge: Cambridge University Press, 2019).
12. Dasgupta, S. and Hill, B. Scratch Community Blocks: Supporting Children as Data Scientists. in *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems - CHI '17*. (New York: ACM Press, 2017), 3620–3631. doi: 10.1145/3025453.3025847.
13. Denning, P. Remaining trouble spots with computational thinking. *Communications of the ACM* 60, 6 (2017), 33–39.
14. diSessa, A. *Changing minds: Computers, learning, and literacy*. (Cambridge: Mit Press, 2001).

15. diSessa. A. What If Your Project's Timeline is a 100 Years?: Reflections on Computational Literacies. in *Proceedings of the 2017 Conference on Interaction Design and Children*. (New York: ACM, 2017), 1–2.

16. Feurzeig, W., Wexelblat, P, and Rosenberg, R. SIMON-A Simple Instructional Monitor. *IEEE Transactions on Man-Machine Systems*, 11, 4 (1970), 174–180.

17. Fields, D., Vasudevan, V., and Kafai, Y. The programmers' collective: fostering participatory culture by making music videos in a high school Scratch coding workshop. *Interactive Learning Environments*. 23, 5 (2015), 613–633.

18. Fields, D., Kafai, Y., Nakajima, T., Goode, J., and Margolis, J. Putting Making into High School Computer Science Classrooms: Promoting Equity in Teaching and Learning with Electronic Textiles in Exploring Computer Science. *Equity & Excellence in Education* 51, 1 (2018), 21–35. doi: 10.1080/10665684.2018.1436998

19. Franklin, D., Skifstad, G., Rolock, R., Mehrotra, I., Ding, V., Hansen, A., Weintrop, D., and Harlow, D. Using Upper-Elementary Student Performance to Understand Conceptual Sequencing in a Blocks-based Curriculum. in *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education - SIGCSE '17*. (New York: ACM, 2017), 231–236. doi: 10.1145/3017680.3017760.

20. Freire, P. *Pedagogy of the oppressed*. (USA: Bloomsbury Publishing, 2018).

21. Giannakos, M., Jaccheri, L., and Proto, R. Teaching computer science to young children through creativity: Lessons learned from the case of Norway. in *Proceedings of the 3rd Computer Science Education Research Conference on Computer Science Education Research*. (2013), 103–111.

22. New London Group. A pedagogy of multiliteracies: Designing social futures. *Harvard educational review*. 66, 1 (1996), 60–93.

23. Grover, S. and Basu, S. Measuring Student Learning in Introductory Block-Based Programming: Examining Misconceptions of Loops, Variables, and Boolean Logic. in *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education - SIGCSE '17*. (New York: ACM Press, 2017), 267–272. doi: 10.1145/3017680.3017723.

24. Grover, S., Basu, S., and Schank, P. What we can learn about student learning from open-ended programming projects in middle school computer science. in *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*. (New York: ACM Press, 2018), 999–1004.

25. Grover, S. and Pea, R. Computational Thinking in K–12: A Review of the State of the Field. *Educational Researcher* 42, 1 (2013), 38–43. doi:10.3102/0013189X12463051.

26. Haraway, D. Situated knowledges: The science question in feminism and the privilege of partial perspective. *Feminist studies*. 14, 3 (1988), 575–599.

27. Harel, I. *Children designers*. (Norwood, NJ: Ablex, 1990).

28. Hautea, S, Dasgupta, S., and Hill, B. Youth Perspectives on Critical Data Literacies. in *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems - CHI '17*. (New York: ACM Press, 2017), 919–930. doi: 10.1145/3025453.3025823.

29. Ito, M., Gutiérrez, K., Livingstone, S., Penuel, B., Rhodes, J., Salen, K., Schor, J., Sefton-Green, J., and Watkins, S. *Connected Learning: An agenda for research and design*. (Cork: Digital Media and Learning Research Hub, 2013).

30. Jacob, S., and Warschauer, M. Computational Thinking and Literacy. *Journal of Computer Science Integration*. 1, 1 (2018). doi: 10.26716/jcsi.2018.01.1.1

31. Jamshidi, F. and Marghitu, D. Using Music to Foster Engagement in Introductory Computing Courses. in *Proceedings of the 50th ACM Technical Symposium on Computer Science Education (SIGCSE '19)*. (New York: ACM Press, 2019), 1278–1278. doi: 10.1145/3287324.3293855.

32. Jayathirtha, G., Kafai, Y., Lui, D., Shaw, M.,and Cho, J. Collaborative Coding and Composing of JazzHands: Integrating the Learning of Advanced Computational Concepts with Electronic Textiles to Make Music Wearables. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*. (New York: ACM Press, 2019), 1274–1274.

33. Kafai, Y. *Minds in Play*. (Norwood, NJ: Ablex, 1995).

34. Kafai, Y. From computational thinking to computational participation in K–12 education. *Communications of the ACM*. 59, 8 (2016), 26–27.

35. Kafai, Y., Fields,D., Lui, D., Walker, J., Shaw, M., Jayathirtha, G., Nakajima, T., Goode, J., and Giang. M. Stitching the Loop with Electronic Textiles: Promoting Equity in High School Students' Competencies and Perceptions of Computer Science. in *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*. (New York: ACM Press, 2019), 1176–1182.

36. Kafai, Y. and Vasudevan, V. Constructionist Gaming Beyond the Screen: Middle School Students' Crafting and Computing of Touchpads, Board Games, and Controllers. In *Proceedings of the Workshop in Primary and Secondary Computing Education on ZZZ - WiPSCE '15*. (New York: ACM Press, 2015), 49–54. doi: 10.1145/2818314.2818334.

37. Lachney, M., Babbitt, W., and Eglash, R. Software design in the 'construction genre' of learning technology: Content aware versus content agonistic. *Computational Culture: A Journal of Software Studies* 5 (2016).

38. Lee, C. and Garcia, A. "I Want Them to Feel the Fear…": Critical Computational Literacy as the New Multimodal Composition. in *Exploring Multimodal Composition and Digital Writing*, edited by Ferdig, R. and Pytash, K. (Hershey, PA: Information Science Reference, 2014), 364–378.

39. Lee, C. and Soep, E. None But Ourselves Can Free Our Minds: Critical Computational Literacy as a Pedagogy of Resistance. *Equity & Excellence in Education*. 49, 4 (2016), 480–492. doi: 10.1080/10665684.2016.1227157.

40. Maloney, J., Peppler, K., Kafai, Y., Resnick, M., and Rusk, N. Programming by Choice: Urban Youth Learning Programming with Scratch. in *Proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education (SIGCSE '08)*. (New York: ACM Press, 2008), 367–371. doi: 10.1145/1352135.1352260.

41. Margolis. J. *Stuck in the shallow end: Education, race, and computing*. (Cambridge: MIT Press, 2008).

42. Margolis, J. and Fisher, A. *Unlocking the clubhouse: Women in computing*. (Cambridge: MIT Press, 2003).

43. Margolis J., Ryoo, J., Moreno Sandoval, C. D., Lee, C., Goode, J., and Chapman, G.. 2012-12. Beyond Access: Broadening Participation in High School Computer Science. *ACM Inroads*. 3, 4 (2012-12), 72–78. doi: 10.1145/2381083.2381102.

44. Morrell, E. *Critical literacy and urban youth: Pedagogies of access, dissent, and liberation*. (Routledge, 2015).

45. Nelson, G. and Ko, A. On Use of Theory in Computing Education Research. in *Proceedings of the 2018 ACM Conference on International Computing Education Research (ICER '18)*. (New York: ACM Press, 2018), 31–39. doi: 10.1145/3230977.3230992.

46. Palumbo, D. Programming Language/Problem-Solving Research: A Review of Relevant Issues. *Review of Educational Research*. 60, 1 (1990), 65-89.

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

48. Paris, D. Culturally sustaining pedagogy: A needed change in stance, terminology, and practice. *Educational Researcher* 41, 3 (2012), 93–97.

49. Pea, R. and Kurland, D. On the cognitive effects of learning computer programming. *New ideas in psychology* 2, 2 (1984), 137–168.

50. Proctor, C., Bigman, M., and Blikstein, P. Defining and designing computer science education in a K-12 public school district. in SIGCSE '19: *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*. (New York: ACM Press, 2019). 314-320. doi: 10.1145/3287324.3287440.

51. Proctor, C. and Blikstein, P. How Broad is Computational Thinking? A Longitudinal Study of Practices Shaping Learning in Computer Science. in *Rethinking learning in the digital age: Making the Learning Sciences Count. 13th International Conference of the Learning Sciences*. (International Society of the Learning Sciences, 2018), 544–551.

52. Proctor, C. and Blikstein, P. Unfold Studio: Supporting critical literacies of text & code. *Information and Learning Science* 1, 2 (2019).

53. Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., and Silverman, J. Scratch: Programming for all. *Communications of the ACM*. 52, 11 (2009), 60–67.

54. Rich, P. and Hodges, C. *Emerging research, practice, and policy on computational thinking*. (Springer, 2017).

55. Richard, G. and Kafai, Y.Blind Spots in Youth DIY Programming: Examining Diversity in Creators, Content, and Comments Within the Scratch Online Community. In Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16). (New York: ACM Press, 2016), 1473–1485. doi: 10.1145/2858036.2858590.

56. Rode, J., Weibert, A., Marshall, A., Aal, K., von Rekowski, T., El Mimouni, J., and Booker, J. From Computational Thinking to Computational Making. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp '15)*. (New York: ACM Press, 2015), 239–250. doi: 10.1145/2750858.2804261.

57. Sandoval, W. Conjecture mapping: An approach to systematic educational design research. *Journal of the learning sciences* 23, 1 (2014), 18–36.

58. Scribner, S. Literacy in Three Metaphors. *American Journal of Education* 93, 1 (1984), 6–21. doi: 10.1086/443783.

59. Searle, K., Fields, D., and Kafai, Y. Is sewing a "girl's sport"? Addressing gender issues in making with electronic textiles. *Makeology: Makers as Learners* 2 (2016), 72.

60. Searle, K. and Kafai, Y. Boys' Needlework: Understanding Gendered and Indigenous *Perspectives on Computing and Crafting with Electronic Textiles. In Proceedings of the Eleventh Annual International Conference on International Computing Education Research (ICER '15)*. (New York: ACM Press, 2015), 31–39. doi: 10.1145/2787622.2787724.

61. Sfard, A. On two metaphors for learning and the dangers of choosing just one. *Educational researcher*. 27, 2 (1998), 4–13.

62. Solomon, C. *Computer environments for children: A reflection on theories of learning and education*. (Cambridge: MIT press, 1988).

63. Soloway, E. and Spohrer, J. *Studying the novice programmer*. (Psychology Press, 2013).

64. Tedre, M. and Denning, P. The long quest for computational thinking. in *Proceedings of the 16th International Conference on Computing Education Research*. (New York: ACM Press, 2016), 120–129.

65. Tenenberg, J. and Knobelsdorf, M. Out of our minds: a review of sociocultural cognition theory. *Computer Science Education*. 24, 1 (2014), 1–24.

66. Tissenbaum, M, Sheldon, J., and Abelson, H. From Computational Thinking to Computational Action. *Communications of the ACM* ,62, 3 (2019), 34–36. doi: 10.1145/3265747.

67. Vakil, S. A Critical Pedagogy Approach for Engaging Urban Youth in Mobile App Development in an After-School Program. *Equity & Excellence in Education*. 47, 1 (2014), 31–45. doi: 10.1080/10665684.2014.866869.

68. Vakil, S. Ethics, Identity, and Political Vision: Toward a Justice- Centered Approach to Equity in Computer Science Education. *Harvard Educational Review* 88, 1 (2018-03), 26–52. doi: 10.17763/1943-5045-88.1.26.

69. Vee, A. *Coding literacy: How computer programming is changing writing*. (Cambridge: Mit Press, 2017).

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

71. Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., and Wilensky, U. Defining Computational Thinking for Mathematics and Science Classrooms. *Journal of Science Education and Technology*. 25, 1 (2016), 127–147. doi: 10.1007/s10956-015-9581-5.

72. Weintrop,D., Killen, H., Munzar, T., and Franke, B. Block-based Comprehension: Exploring and Explaining Student Outcomes from a Read-only Block-based Exam. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education (SIGCSE '19)*. (New York: ACM Press, 2019), 1218–1224. doi: 10.1145/3287324.3287348.

73. Weintrop, D. and Wilensky, U. 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)*. (New York: ACM Press, 2015), 101–110. doi: 10.1145/2787622.2787721.

74. Werner, L., Denner, J., and Campe, S. Children Programming Games: A Strategy for Measuring Computational Learning. *Transactions of Computing Education*. 14, 4 (2014), 1–22. doi: 10.1145/2677091.

75. Wing, J. Computational Thinking. *Communications of the ACM*. 49, 3 (2006), 33–35. doi: 10.1145/1118178.1118215.

76. Zaidi, R., Freihofer, I., and Townsend, G. Using Scratch and Female Role Models While Storytelling Improves Fifth-Grade Students' Attitudes Toward Computing. in *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education (SIGCSE '17)*. (New York: ACM Press, 2017), 791–792. doi: 10.1145/3017680.3022451.

Yasmin Kafai
Graduate School of Education
University of Pennsylvania
3700 Walnut Street, Philadelphia, PA 19104
kafai@upenn.edu

Chris Proctor
Graduate School of Education
Stanford University
485 Lausen Mall, Stanford, CA 94305
cproctor@cs.stanford.edu

Debora Lui
Graduate School of Education
University of Pennsylvania
3700 Walnut Street, Philadelphia, PA 19104
Ablui@gmail.com

From *Proceedings of the 2019 ACM Conference on International Computing Education Research.* Reprinted with permission.
DOI: http://dx.doi.org/10.1145/3381887