This paper was commissioned for the Committee on Enhancing Science and Engineering in Prekindergarten through Fifth Grades, whose work was supported by the Carnegie Corporation of New York and the Robin Hood Learning + Technology Fund. Opinions and statements included in the paper are solely those of the individual authors, and are not necessarily adopted, endorsed, or verified as accurate by the Committee on Enhancing Science and Engineering in Prekindergarten through Fifth Grades, the Board on Science Education, or the National Academy of Sciences, Engineering, and Medicine.

The Integration of Computational Thinking in Early Childhood and Elementary Science and Engineering Education[1]

Diane Jass Ketelhut and Lautaro Cabrera
University of Maryland College Park

Computational thinking (CT) has been defined as a set of practices emerging from computer science (Wing, 2006), a set of dispositions (Computer Science Teachers Association [CSTA] & International Society for Technology in Education [ISTE], 2011), and a way of thinking (National Research Council [NRC], 2011). Although its boundaries and definitions are often contested (Tedre & Denning, 2016), there is some consensus around CT as a valuable skill for students to engage in an increasingly technological and computational world (Grover & Pea, 2013; NRC, 2011). CT can also be valuable as a method for developing disciplinary understanding (for early examples in math and science, see diSessa, 2000; Papert, 1980). This application of CT is most important in subjects where computational methods and concepts are intertwined with typical disciplinary practices—such as science and engineering. For example, there is a close resemblance between aspects of scientific inquiry and aspects of CT such as data collection and analysis. In engineering, the CT practices of defining problems through abstraction and approaching solutions systematically parallel typical applications of working with robots and testing solutions iteratively. Therefore, we conceptualize CT for the purposes of this paper from a disciplinary perspective: ***as a set of skills necessary to formulate science and engineering problems and their solutions so they can be carried out by a computational agent*** (Lee & Malyn-Smith, 2020; Wing, 2010).

The educational synergy between science, engineering and CT has been explored empirically, mostly at the middle and secondary levels. For example, Weintrop et al. (2016) investigated how professional scientists used computational practices in their work and developed a taxonomy of CT practices to be integrated into math and science at the high-school and college levels. Within K-12 education, Sengupta et al. (2013) studied how middle-school students explored curricular units of physics and biology by engaging with computational agent-based modeling. Additionally, Basu et al. (2016) explored how block-based programming environments can serve as mediums for learning and assessment of scientific content. Other research connects robotics with engineering design at the secondary school level (Grover, 2011). Overall, researchers have found that CT and computational environments can be particularly productive methods to learn about science, engineering, and math—a finding that builds on research predating Wing's (2006) popularization of CT as a term (diSessa, 2000; Papert, 1980). However, the integration of CT into science and engineering education at the **elementary** level is still a burgeoning field despite the fact that these years are key for exposure and in the development of academic interest, confidence and proficiency. If we are to prepare all students to participate in increasingly computational disciplines and society, we should provide foundational computing experiences for students during these important years.

Goals for integrating computational thinking into elementary science and engineering education

The integration of CT into science and engineering education at the elementary level has three **equally important** main goals that should happen simultaneously.

1. Provide early experiences with computing to develop foundational proficiency in computing;
2. Increase equity of access to computing education;
3. Advance science and engineering understanding.

[1] This paper responds to our charge to write a "paper that reviews the empirical research on the integration of computational thinking into early childhood and elementary science and engineering education. This paper should also include the author's interpretation of the literature, including the broad conclusions that can be drawn from and supported by the extant research, and aspects that merit further study."

Goal 1: In order for students to feel comfortable engaging with computing at the middle and high-school level, they need a foundation in the concepts and processes of computing (Association for Computing Machinery, 2016). As science and engineering become more computational at the professional level, it becomes increasingly necessary to understand computing principles and how they can be used in disciplinary contexts (Malyn-Smith et al., 2017). As indicated above, there is a natural synergy between CT and science and engineering. Integrating CT at the elementary level can provide students with the necessary proficiency, interest, and confidence in computing to engage in computational activities within these disciplines.

Goal 2: The first aim is also closely related to this second goal of why to integrate CT at the elementary level: to promote equity in computing education. While it is important to provide students with the necessary foundational experiences in computing, it is equally important to make those opportunities available to *all* students and to consider how students from different backgrounds can engage in computational thinking (Santo et al., 2019). Integrating CT at the elementary level has the potential to reach students at a young age when academic interests are early in development, making it an ideal context to reach students that are typically underrepresented in computing education. However, it is critical that educators adapt the teaching of CT in science and engineering to the specific academic and cultural backgrounds of their students (Ryoo, 2019).

Goal 3: The third goal of integrating CT is in service of disciplinary learning: to leverage the power of computing to advance science and engineering understanding (Lee et al., 2020). It is no accident that professional sciences and engineering are increasingly computational—computing provides new tools and methods to investigate topics with more speed, at a larger scale, and in some cases, in a more engaging or intuitive way. Therefore, integrating CT into science and engineering can take advantage of new ways to engage students in science and engineering, allowing them to explore topics in authentic, engaging, and in-depth ways.

Our Purpose

In this paper, we aim to (a) examine how and with what effects CT has been integrated into elementary science and engineering education, (b) illustrate ways of integrating CT at this level and offer example cases, (c) identify gaps in CT integration research and practice, and (d) propose ways for educators and researchers to further advance the goals of CT integration.

However, we also impose some constraints on our analysis to focus the conversation. First, we center our discussion on CT integration in the United States, as a myriad of local factors such as widely adopted standards (NGSS), racial and gender equity issues in STEM, and government focus on science and engineering education (National Science and Technology Council, 2018; NSF, NAS) make international comparisons less informative. However, we have referenced studies from international venues to highlight what is possible where US examples are missing and what is developmentally appropriate for students at the elementary level.

Second, we strive to base our analysis and recommendations on peer-reviewed empirical evidence, while understanding that CT is a relatively nascent strand of research in education. Although computing education has a richer history dating back to the 1960s and there is foundational work pertinent to our goals in this paper (e.g., Papert 1980; diSessa 2000), we focus on research that investigates the integration of CT into science and engineering education in the last couple of decades as the role and tools of computing education have shifted immensely in the last 20 years.

Third, we limit our analysis to cases of CT *integration*—paying less attention to examples of computer science or CT learning that are not associated with disciplinary content. While we believe those efforts are important and informative to the larger field of computing education, they are more pertinent to other

analyses that focus on computer science education at the K-12 level. This limitation has resulted in all of our examples being from the last 10 years.

CT Integration

To organize our analysis on the integration of CT into science and engineering education, we use Waterman et al.'s CT integration framework (2019), which describes three levels of integration: Exist, Enhance, and Extend. At the Exist level, integration entails first, highlighting current practices in the curriculum that resemble CT practices to teachers, and second, suggesting that the teachers point these out or make them explicit to their students. At the Enhance level, additional activities are integrated into typical science and engineering lessons, sometimes through substitution of one lesson or activity, to provide an explicit connection to computing—usually in a basic or foundational way. At the Extend level, new lessons are added to a unit that involve computing—typically programming or modeling—to advance the science and engineering content knowledge of the unit while also increasing computational understanding.

Below, we use Waterman et al.'s framework (2019) to organize the literature on the integration of CT into elementary science and engineering education. At each level of integration, we describe patterns and themes in the literature; share examples of integration in different grades and disciplines where they exist; indicate how the goals of CT integration into science and engineering are promoted and addressed (or not); and end with a discussion of the affordances and challenges of integrating CT at that level.

Overall, there are not a lot of empirical articles that investigate the integration of CT with science and/or engineering at the PreK-5 grade levels, and they all stem from 2010-2020. There are a number of *projects* that are investigating this area, but they are apparently not mature enough to have empirical publications. Interestingly, the articles we analyzed were split at both ends of the integration spectrum: the Exist and Extend levels. Only one international study represented an Enhance level of integration.

In our analysis, we determined the integration level of a project iteratively by discussing each study and how its attributes matched the definitions of each integration level. In some examples below, we explain how *aspects* of each project may individually fall into different integration levels, but we made our final integration level judgements based on each unit as a whole. Therefore, this analysis shows a suggestive state of the field based on our observations rather than a precise categorization based on a fixed set of criteria. Table 1 shows a summary table, organized by integration level.

Table 1. Summary of published student-focused empirical studies on CT integration into PreK and elementary science and engineering

Citation	Location	Grade/Age	Participants information	Science content	CT focus	Integration level
Dwyer et al., 2014	In school	4th	From a diverse school (no individual data)	Physics	Unplugged programming	Not integrated[2]
Gürbüz et al., 2017	Turkey	8-10 year olds	Previous experience	Weather	Algorithmic thinking	Not integrated
Leonard et al., 2015	In school	5th	High percent white/gifted	Cell biology (mixed with dance)	Programming	Not integrated
Luo et al., 2020	Summer camp	1st-5th	Two girls: one Asian-American and one White	Biology	Programming	Not integrated
Pinto-Llorente et al., 2018	Spain: magnet school	4th	Highly experienced	Simple/complex machines	Programming	Not integrated

[2] Studies categorized as "not integrated" do not fall into Waterman's framework. These are explained in a section after the extend section.

Toma et al., 2019	Spain	5th	Rural	Inquiry science	Coding	Not integrated
Ehsan et al., 2020	Science Center	5-7 year olds	Primarily white boys	Engineering Design	Broad	Exist
Hynes et al., 2016	In school	K	No information	Engineering	Broad	Exist
Mensan et al., 2020	Malaysia	5th	Rural, majority girls	Matter	Unplugged, decomposition, abstraction, pattern recognition, algorithmic thinking	Exist
Kalogiannakis et al., 2018	Greece, summer school	K-2nd	No information	Gravity	Programming	Enhance
Basu et al., 2015	In school	5th	No information	Ecosystems, Kinematics	Modeling; Programming	Extend
Dickes et al., 2019	In school	3rd-5th	Diverse racially; urban, mixed gender	Ecosystems	Modeling, programming	Extend
Horn et al., 2014	Museum	9-16 year olds	Not reported	Evolution	Programming	Extend
Sengupta & Farris, 2012	Outside school	3rd-4th	Mostly white boys	Kinematics	Computational modeling	Extend

CT at the Exist Level

At this lowest level of integration, researchers and educators aim to simply "call out" or make explicit CT practices that are already embedded in the curriculum or standards but are not identified as computational. In the context of science learning, researchers often make the argument that the practices of data collection, analysis, and graphing—typically associated with scientific inquiry practices—are examples of CT that already exists in lessons. For example, in our own work, we have seen teachers identify the measurement of plant growth as an example of 'unplugged' (not using computers or other technology) CT data collection and analysis. This conflation between CT and scientific practices can be confusing to teachers, and possibly result in a lack of instructional change (Ketelhut, Mills, et al., 2019).

In engineering contexts, the practices of iterative design and troubleshooting are often associated with the CT practices of debugging and problem decomposition. For example, Ehsan and colleagues (2020) created an engineering design exhibit ("build a puppy play yard") at a family science center. They then analyzed the actions of ten 5-7 year old children, primarily white boys, as they interacted with this exhibit to see if they demonstrated computational thinking. As one example, the authors identified the CT skill of abstraction when a child said they would build something for the puppy to play with and add a fence in response to the repeated parent question of what they will build. Others studies did something similar: look at curriculum or children's behavior and then, map it onto CT practices.

Example Box 1 offers another look into an engineering design Exist project that is unplugged. In this example, the researcher/designers analyzed an already-designed five session engineering design project to see what aspects could map to CT. They then observed a Kindergarten teacher and her students engaging with this project. We classify this as an Exist project because the unit was already designed as an engineering design project, and the teacher made no connections to CT in teaching it. However, we recognize that this unit could easily make the jump to an Enhance level of integration with some modification. For example, they could try to write their treasure map in a way that a robot, who can only understand very precise instructions, can follow. Or, they could try to write the process of answering yes/no questions in a flowchart, making decisions about which questions should be asked first to discover the answer with the least amount of questions possible (connecting to the concept of efficiency in computing).

Example 1: Build a Toy Box (Hynes et al., 2016)—Grades K-2
Session 1: Students are introduced to the design project, and develop needs and issues; they engage in an activity on developing a treasure map to underscore the need for a standardized measurement (learning that "paces" are not standard).
Session 2: Students continue to explore measurement tools and options.
Session 3: Students investigate and learn about physical properties, using the book *Living Color*. They attempt to discover the identity of a hidden object using yes/no questions about its properties.
Session 4: Students plan out their toy box and test out materials to use in its construction.
Session 5: Students build and test their toy box. They share with the class and discuss possible redesigns, which they then implement.

While lessons that apply CT integration at the Exist level can be an important stepping stone towards more intricate computing applications, their value in meeting the three goals ascribed to CT integration is less clear. For instance, if we accept that CT integration should help students develop proficiency in computing, is calling out existing CT practices in otherwise traditional disciplinary instruction developing a new type of skill? In terms of content integration, do students come to understand science and engineering topics in a *different* way if educators simply highlight overlaps between typical instruction and CT practices? The answer to those questions is unknown as the research base is simply too immature to answer them. However, we hypothesize that it is crucially important that children know that they are engaging in CT practices for these goals to be achieved. If children do not know that they are engaging in computing, is there any reason to believe they will develop an interest in computing as a result of these lessons? Or, apply whatever they learn in these lessons when they encounter a computational environment? Moreover, how does pointing out existing practices as CT contribute to expanding participation in computing? If the answer to this is no, then relying on exist level projects for children undermines the goal for expanding participation. Further, in our review and experience, these Exist level projects tend to be more often associated with younger grades and with engineering design, as our example above indicates, limiting important years to develop interest in computing even more.

In our view, it seems unlikely that Exist level integration can significantly contribute to these goals. However, we value these efforts as ways of getting educators comfortable with CT and develop the competencies and confidence to venture further into computing integration. If teachers can see where something is 'like CT', then they can move to the next level, Enhance, of adding or substituting a CT activity directly. As Waterman et al. (2019) argued, finding the CT that already exists in the curriculum can help teachers identify opportunities for more advanced CT integration **if they are supported in doing so**.

CT at the Enhance Level

At this second level of integration, additional lessons or activities are added to typical science and engineering instruction. These lessons focus on advancing disciplinary understanding while explicitly highlighting connections to computing practices. In our review of the literature, we found few examples of integration at this level. The exception is a study in Greece by Kalogiannakis et al (2018) in integrating CT into a unit on gravity for 5-7 year olds in summer school. Example Box #2 illustrates their implementation. It should be noted that the teacher received CT professional development (PD) prior and became part of the research design team in creating this unit.

Example 2: ScratchJr and Gravity (Kalogiannakis et al., 2018)—Grades K-2
Activity 1*: Students investigate what objects can roll down a ramp; interactive and inquiry-based.
Activity 2: Students explore the impact of changing the height of the ramp on how far an

object can roll.
Activity 3: Students observe and hypothesize about paint dripping down a piece of paper.
Activity 4: Students watch teacher-designed ScratchJr scenarios about gravity.
Activity 5: Students also engage with other interactive multimedia.
Activity 6: Students reflect, design and experiment on gravity related to their own questions with access to online informational sites.
Session 7 & 8: Students create their own ScratchJr program about what they understand about gravity.

*Activities are presented sequentially as best as can be interpreted from the manuscript; however, some of these activities might be more integrated than appears here.

It's important to note that, while this unit includes some ScratchJr programming, we categorize it as Enhance because the ScratchJr activities are at a basic level and not meant to provide students with additional ways to *explore* the content, but another way to demonstrate their understanding of it. Because the students are creating a simple animation to show their knowledge, there is no synergy between the computational aspects of the assessment and the disciplinary content. For example, students can show their understanding of gravity by "making things fall" through an A-to-B animation, but that "fall" is unrelated to the programming blocks used for the animation. On the other hand, the integration of CT would be richer if students used the programming environment to create their animation based on relationships of force and acceleration—which is likely too complex for second graders but doable with upper-elementary students (diSessa, 2000).

The fact that the teacher was integrated into a team of researchers and designers to create this unit might partially explain the paucity of examples at this level of integration. This type of unit requires that the teacher has a strong balance of disciplinary and computing knowledge. For educators to highlight overlaps between two fields of knowledge for students, they require a deep understanding of both disciplines and their connections. Given that most teachers have little to no background in computing education and that elementary school teachers are by training generalists who seldom have the option to specialize in science or engineering teaching (Horizon Research, 2019), achieving the type of balance that Enhance level integration requires seems problematic. Elementary teacher education programs are already crammed with courses to help teachers develop the disciplinary and pedagogical knowledge to teach all subjects. Increasing science credits and adding computer science ones would require something else to give. As we discuss below, the solution might be to create integrated science/computer science methods courses that model Enhance level integration. However, those too run into the issue of whether University-based science methods teacher educators have the required combined level of expertise. This difficulty is exacerbated by the need to develop expertise in science and CT content as well as in pedagogical approaches in both of those disciplines.

CT at the Extend Level

At this final level of integration, CT is an integral part of lessons and activities; the *method* for exploring a scientific or engineering concept. In many cases, this integration is enacted through programming—students use a developmentally appropriate programming environment to create models, test scenarios, and design solutions within disciplinary topics. For example, Dickes et al. (2019) created a fifteen 50-minute-lesson unit where students explored an ecosystem within an immersive virtual environment. Students also engaged with a 2D agent-based modeling environment where they use programming to control the behaviors of animals in the environment as see the outcomes in the ecosystem (see Example Box #3). The authors demonstrate different moments of "transformative modeling" where students transform the disciplinary content from one type of representation to another. Overall, this extensive implementation

resulted in students advancing their understanding of both the scientific concepts of the curriculum and the purpose and mechanisms of computational models.

> **Example 3: Ecosystems and Computational Modeling** (Dickes et al., 2019)—Grades 3-5
> Session 1: Students are introduced to the virtual immersive environment and the science problem.
> Session 2-3: Students explore the environment, time traveling to past years, and begin to collect data.
> Session 3-6: Students are introduced to the block-based programming modeling tool; they create 'beaver point of view' models.
> Session 7: Students explore the environment, time traveling to a more recent year, and collect data.
> Session 8-9: Students are introduced to the woodpecker point of view and mapping tools.
> Session 10: Students create concept maps; they explore the current environment.
> Session 11-12: Students create woodpecker point of view models in the programming environment.
> Session 13-14: Students synthesize, graph and make causal maps.
> Session 15: Students share.

While this example shows the most intricate integration of CT, it also demonstrates the requirement of extensive support to achieve the Extend level. In the case we highlighted, researchers had designed computational modeling tools, trained teachers in using these tools with professional development, and supported teachers through integration. They also counted on curricular flexibility and extended time to implement their innovation. While the amount of type of support varied among Extend projects, in general they all required additional help beyond the teacher.

At the Extend level, then, the integration of CT is clearly impacting the first and third goal we proposed above: students are becoming more proficient with computing and advancing their disciplinary understanding. However, the contribution towards the goal of expanding computing access is less clear. In the example we showed which is not atypical for the group, most students had previous experiences with gaming; and half had experience with programming. These previous experiences, coupled with the high requirements of support, specialized materials, and instructional time, put into question the feasibility of Extend level integrations in contexts where students have different background experiences, teachers have no access to modeling tools designed for their curriculum, or support from a CT researcher is unavailable. To be clear, these limitations do not dampen the potential of Extend level integrations—this highest level should be our high aspiration. But, when considering the integration of CT across the vast landscape of K-5 schools realistically, Extend projects seem difficult to manage at scale.

Another pattern we found in the studies at this level of integration is that implementations were geared towards upper elementary levels (3^{rd}- 5^{th} grades). While, at first glance, one could argue that this focus on higher levels reflects the definition of the Extend level's requirements for programming or computational modeling, the evidence from the body of literature at large does not support that contention. A large percent of *all* the studies we examined integrated programming practices, and they spanned all elementary grades. So, the focus of Extend projects on upper elementary grades cannot be solely explained by the CT practices chosen to be integrated. It is unclear whether the focus of Extend projects on upper elementary is a meaningless anomaly due to the lack of a robust body of literature or due to another reason, such as the developmentally appropriateness of investigating disciplinary topics in authentic computing environments.

CT Disjointed (not integrated)

Not all the studies we found that purport to integrate CT into science and engineering fall within Waterman

et al.'s framework (2019). Integration, by definition, indicates bringing objects together to create a cohesive whole, whereas nearly half of our studies *interjected* CT activities—specifically programming activities—into typical science or engineering lessons without making clear connections between CT and disciplinary content. In these studies, CT learning was not instrumented as a tool to learn disciplinary content. Instead, science or engineering served as the context for students to engage in activities aimed at improving their CT skills. We call those studies "disjointed", and they represented half of the published empirical articles that we found.

In all of these projects, programming at some level was added into a science or engineering unit from Kindergarten to 5th grade with little or no connection to the science or engineering content, beyond that of context. In some cases, the article simply did not explain how the programming lessons were connected to the content. The most popular approach across the grades was to use a form of robotics (e.g. Lego Education WeDo in the Pinto-Llorente et al. [2018] study). Two studies did not make their approach clear and a third used a visual block-based programming environment for 5th graders.

Luo et al. (2020) is a strong example in this category approaching integration. They followed two girls (3rd grade, Asian-American and 5th grade White) through a four week unit in summer camp. This eight-session unit was to integrate programming robots with learning life cycles of ferns and mosses. See the details in Example Box #4.

> **Example 4: Robots and Ferns/Mosses** (Luo et al., 2020)—Grades 3-5
> Session 1: Introduction to robots and coding in general and specifically the Dash robot and Blockly software. Learned how to move the robot.
> Session 2: Investigated the science of ferns and created a sequence of the life cycle of the fern in what the researchers called an unplugged activity.
> Session 3: The paper sequences of the fern life cycle are inputted into the robot using several sound blocks.
> Session 4: Session 2 is repeated for mosses with the variant of introducing the concept of loops.
> Session 5: Repeat session 3 with the goal of using the sound blocks to discuss the differences between the fern and moss life cycles. Loops are included by having the robot repeat twice.
> Session 6: Introduce conditionals and continue on the differences between the fern and moss life cycles.
> Session 7 & 8: Open coding exploration unrelated to ferns and mosses.

In this unit, some sessions are dedicated exclusively to coding (e.g., 1, 7, and 8) but those lessons do not position coding as a way to interact with the science content—they are devoid of disciplinary connections. Other sessions introduce the science as a context for a programming exercise (e.g., 3 and 5).

How do the disjointed studies address our three goals? With their emphasis on coding, these studies as a whole are preparing students to engage with computing at the middle and high-school level by providing a foundation in the concepts and processes of computing (Goal 1). Their support for our Goal 2 is less clear. Goal 2 is to promote equity in computing education. Sadly, over half of these students and/or schools were intentionally chosen because of their prior knowledge in computer science or learning technologies—which seems to go against the goal of *expanding* computing education. In terms of Goal 3, because of their separation of science and/or engineering content and CT, these interventions would be unlikely to support increased science and engineering understanding. The lack of connection between computing activities and disciplinary content prevents students from benefitting from CT as a way to further science or engineering understanding.

Teacher Education

While the existing research investigating how students can engage in CT within science and engineering contexts provides an important foundation to understand how CT can be integrated into K-12 classrooms, there is another key component to realizing full integration: teachers. The integration of CT into science and engineering depends on the ability of educators who can infuse CT into their instruction (Barr & Stephenson, 2011) while maintaining developmental appropriateness and curricular demands; guide and support their students through CT activities in disciplinary contexts; and provide CT learning opportunities for all their students. Clearly, we expect a great deal of labor, enthusiasm, and pedagogical content knowledge (Shulman, 1986) from teachers to achieve CT integration.

Therefore, it is the task of pre-service teacher educators and professional development designers to create opportunities for teachers to develop the necessary skills, confidence, and excitement to promote the integration of CT in their own classrooms. We note, however, that simply understanding how to integrate CT into science does not imply that teachers will be able to or that they will do so equitably. However, without the skills, confidence and excitement around CT, integration is unlikely to happen. In this section, we review existing efforts to prepare pre- and in-service teachers to integrate CT into elementary science and engineering instruction. We describe how different initiatives associate to CT integration levels (Waterman et al., 2019) and discuss remaining questions and future research directions. Table 2 summarizes the various pre-service and professional development programs we investigated. Because only a few studies fall under these constraints, we summarize their integration levels within Table 2.

Table 2. Summary of teacher education efforts to integrate CT into science and engineering.

Citation	Context	CT Integration	Participants information	Integration level modeled in instruction	Integration level teachers implemented
Jaipal-Jamani & Angeli, 2017; 2018	Pre-service Science methods course.	Programming robotics activities.	N=21; 11 women, 10 men. Grades 4-8. Had completed a teaching with tech course, but no CT or robotics experience.	Disjointed: teachers completed programming practices that helped them understand "gears" better, but no connection to curriculum or elementary disciplinary learning goals.	None, only participated in CT activities as learners.
Kaya et al., 2019	Pre-service Science methods course.	Programming and robotics activities.	N=56; 50 women, 6 men. Mean age of 26. No prior experience in CT.	Disjointed/Exist: Participated in programming and robotics activities disconnected from disciplinary curriculum. Only *discussed* connections to NGSS.	None, only participated in CT activities as learners.
Ketelhut et al., 2019	After-school year-long PD with pre- and in-service teachers	Algorithms, problem decomposition, systems thinking, models and simulations	Only in-service. N=13; all women. 9 White, 2 African American, 1 Multiple Races (White, Hispanic/Latina), and 1 Asian American.	Modified Enhance: teachers participated in activities with a heavy computational aspect and basic connections to science.	Exist: teachers designed mostly typical science lessons but described them with CT labels.
McGinnis et al., 2020	Pre-service Science methods course.	Programming robotics, data collection and analysis, modeling.	N=39; Women: 24 White; 5 Asian or Asian-American; 2 Black or African American; 2 Hispanic or Latina; 2 Multiracial. Men: 2 White; 2 Multiracial.	Disjointed/Exist: teachers participated in robotics programming activities and one activity of citizen science where they described practices as CT	Mostly Disjointed and Exist: teachers saw CT as an add-on or as existing in curriculum already
Rich et al., 2020	In-service teachers part of a	Unplugged abstraction, decomposition,	N=8. Demographics of teachers not shared. Teachers' schools	Exist and Enhance: researchers developed tools to find existing CT and	Exist and Enhance: teachers used specific CT

	Researcher-practitioner partnership	debugging, and patterns	were 23-66% non-white, 50-76% free and reduced lunch, 4-19% English language learners.	ways to "make existing CT ideas more explicit"	language to describe activities to students and make connections to computing concepts

Note: This table includes only studies focusing on teachers integrating CT into science and engineering.

Pre-service teacher education. One strand of research within teacher education focuses on the preparation of pre-service teachers to integrate CT into their future classrooms. The shared argument among these projects is that, to create widespread instructional change, tomorrow's teachers should be prepared to lead that change. At the same time, pre-service teacher education is an ideal context for innovation as teacher educators can change existing structures (like university methods courses) to implement new ideas—such as CT integration. As evidenced in Table 2, only a few projects have focused on the specific goal of preparing pre-service teachers to integrate CT into science and engineering instruction at the elementary level.

Jaipal-Jamani and Angeli[3] have studied how elementary pre-service teachers develop CT understanding and self-efficacy in integrating CT into science and engineering by introducing educational robots to their students. Their studies show that pre-service methods course modules focused on learning science and programming concepts through robotics can be effective in increasing teacher self-efficacy, science understanding, and CT knowledge (Jaipal-Jamani & Angeli, 2017, 2018). The authors particularly highlight the potential of pre-service education to develop teachers' CT and disciplinary content knowledge "integrated with the teaching of pedagogy" (2017, p. 187). Similarly, Kaya et al. (2019) have found that a science methods course intervention resulted in increased teacher confidence and interest in coding.

Similarly, part of our work has focused on preparing pre-service teachers to integrate CT into science instruction by creating a CT module in an existing elementary science methods university course. In this intervention, we implemented a 3-class CT unit where pre-service teachers learned about CT concepts, programmed elementary-level robots (Lego Mindstorms and Makeblock Mbots), and designed a CT-infused science lesson as a capstone project. Like the studies mentioned above, we have also found that CT modules can effectively increase teacher self-efficacy and CT understanding (Cabrera et al., 2019, 2020; Hestness et al., 2019; Ketelhut, Mills, et al., 2019). In addition, we have found that, when asked to design CT-infused science lessons, pre-service teachers often integrate CT in general ways that are largely compatible with typical science instruction—meaning they integrate CT at the Exist level.

Our analysis of a professional development series where pre-service and in-service teachers learn together has also shown that novice teachers benefit from the experience of in-service teachers while designing CT-infused lessons (Ketelhut, Hestness, et al., 2019; Killen et al., 2020). They most often lean on more seasoned teachers for issues of curricular alignment or organization while contributing their own domain of technology or CT tools available for the lesson design. In this same context, we have also found that engaging with CT-infused lessons as "students" and co-designing lessons with other educators and researchers can lead to teachers designing lessons with higher levels of integration.

While these studies inform how pre-service teachers *learn* about CT, they do not provide information on teachers' ability, willingness, or opportunities to integrate CT into their classrooms *after* instruction. In the studies we reviewed, the only opportunities to assess levels of CT integration were lesson designs created by teachers. And, even in these cases, the levels of integration on the lesson plans may differ from what gets implemented in the classroom. This focus on what teachers do in their pre-service university classroom makes it difficult to assess how these projects impact instruction in the elementary classroom.

[3] All projects discussed work at least with some elementary teachers but may also include teachers from other grades.

In-service teacher professional development. Another strand on research focuses on preparing in-service teachers to integrate CT into science and engineering. However, this strand is significantly thinner, and also generally lacks any classroom investigations. The exception is a study by Rich et al. (2020) where researchers investigated patterns of CT integration among eight teachers and developed profiles to characterize different ways of conceptualizing and framing CT in science lessons. While this one investigation is promising and shows that teachers can learn about CT and integrate it in productive ways, we need a more robust body of literature to make more generalizable conclusions around teachers integrating CT. However, studies that aim to understand how teachers learn about CT integration into *general* instruction (beyond just science and engineering) can enlighten how different initiatives can lead to different levels of integration.

For instance, for several years Yadav and his team at Michigan State University have studied how pre-service teachers learn about CT in a technology integration course. Although the authors do not focus specifically on the integration of CT into science and engineering, Yadav and colleagues have demonstrated how teachers develop understandings around CT (Yadav et al., 2014), explored strategies to measure this knowledge (Yadav et al., 2018), and studied how teachers implement CT in their classrooms (Rich et al., 2020). Overall, the authors have found that pre-service teachers can effectively learn about the main concepts of CT through technology integration courses, reinforcing the findings from science methods courses analyzed above. Similarly, Mouza et al. (2017) have investigated the introduction of CT in a pre-service technology integration course and found that teachers developed conceptualizations of CT that varied in their generality and association with technology.

Research focused on in-service teachers and CT has often been limited to exploring teachers' perspectives of CT and how they think it could be implemented in classrooms (e.g., Garvin et al., 2019; Sands et al., 2018). But, the few studies focused on in-service teacher *learning* around CT and how they integrate CT into other disciplines or higher grades show that, in addition to understanding the conceptual underpinnings of CT, teachers need administrative and pedagogical support to make integration possible. For example, Israel et al. (2015) found that (a) elementary school teachers benefitted from continuous support throughout the year to integrate CT into their math instruction, and (b) limited instructional time pitted the integration of computing at odds with teaching the required curriculum focused on disciplinary content. This issue of alignment between disciplinary curricula and CT integration is echoed in Bain et al.'s study (2020), which found that science teachers at the high-school level would often frame CT as "in conflict with" disciplinary content (p. 92). These and other barriers can lead to teachers most often "integrating" CT in disjointed ways where lessons focus on either disciplinary content or CT learning (Israel & Lash, 2019).

Integration levels. When analyzing these teacher education studies through Waterman et al.'s framework, we must make an important distinction between the level of integration that *teachers* promoted and the level of integration that teacher educators or PD designers *modeled* for them. The criteria for what constitutes CT integration differed across different studies. In most studies, researchers modeled Exist or disjointed levels of integration for teachers. For example, Kaya et al. (2019) described their intervention where they asked pre-service teachers to solve puzzles and robot programming challenges, leaving the "integration" to a discussion of how CT—which had been modeled in isolation through those activities—could be infused into NGSS lessons.

Other studies pushed for higher levels of CT integration and expected teachers to integrate CT at Enhance levels (where they go beyond typical instruction) or higher. For example, Rich et al., (2020) acknowledged that CT could already exist in the classroom but developed specific cognitive tools to support teachers in "mak[ing] existing CT ideas more explicit or embed[ding] new opportunities for CT" (p. 3166).

This progression from Disjointed and Exist levels of integration towards Enhance is also reflected in our work. During the first year of the project, we took a similar approach to most studies reviewed here: we

modeled mostly CT activities for teachers and left the "integration" largely to participants (Hestness et al., 2018). However, after analyzing our first-year data and redesigning the professional development learning environment, we implemented a model where teachers engaged with *already CT-integrated* science activities and then codesigned lessons with researchers and peers. In this case, *we* modeled activities more aligned with Enhance and Extend levels, but saw variability in how *teachers* integrated CT: some remaining at the Exist level and others going beyond (Cabrera et al., in preparation; Ketelhut, Cabrera, et al., 2019).

Overall, studies in teacher education and CT integration show that teachers may develop different conceptualizations of CT and design lessons with varied levels of integration. However, the few systematic investigations of teachers' integration levels have focused on their lesson *plans*, without assessing how (or whether) those plans are implemented in the classroom (Israel & Lash, 2019; Ketelhut, Mills, et al., 2019; McGinnis et al., 2020; for an exception, see Rich et al., 2020). Additionally, it is unclear which kinds of teacher education initiatives lead to variability in integration levels. We propose future research directions below.

Discussion of the state of the field

So far, we have examined how CT is integrated in elementary science and engineering education in K-12 and teacher education. Now, we summarize the state of the field by returning to the three goals of integration.

Goal 1: Providing early experiences in computing to facilitate developing proficiency in computing later.

In our analysis, this first goal was met by a variety of approaches. On one hand, studies that interjected disjointed CT activities still provided early experiences in computing for students. The same was true for teacher education studies: even those that only showed isolated CT activities for teachers were able to increase teachers' self-efficacy around CT and understanding of computing concepts.

However, our review also found variability in "what counts" as an early experience in computing. In some cases, early experiences were considered to be developmentally appropriate versions of programming, such as applications of block-based programming. In other cases, researchers did not involve computational environments in providing students with early computing experiences. For example, studies at the Exist level barely made any changes to typical science or engineering instruction, meaning that students did not engage in any *new* experiences in computing, even though CT had been deemed to be "integrated."

This elasticity in conceptualizing what constitutes early computing experiences was especially evident in studies involving unplugged activities, where teachers implemented lessons that engaged students in thinking strategies *analogous* to what they would need if interacting with a computer (e.g., Mensan et al., 2020). In these studies, the main argument is that these unplugged activities can teach students generalizable thinking strategies that can prepare them to later interact with computing environments. However, this assumption of skill generalizability or transfer is untested—we have few (if any) empirical investigations on the effects of unplugged activities on subsequent computing proficiency. One recent exception showed that students in an experimental condition who participated in an unplugged CT unit performed better at a CT test (Mensan et al., 2020). However, the test was *also* unplugged, showing that students had indeed developed the skills of applying their learning on paper, but not necessarily demonstrating proficiency in computing. Further, these studies show great variability in which CT practices get integrated into which science and engineering topics. Currently, there is no systematic approach to choosing what science to match with what CT skill. Some authors argue for why they think a specific CT skill, like computational modeling, can be particularly productive for science education (Sengupta et al., 2013), but there is no investigation into which science and engineering topics are most fertile for computing integration at the elementary level (for high-school and college grades, see Weintrop et al., 2016). We need to understand

better what aspects of science and engineering can be productive stepping-stones for students to engage in CT and facilitate the development of computing proficiency.

In our view, while unplugged activities *may* provide students with early computing experiences that develop computational proficiency, we should strive to provide more authentic opportunities for students, particularly those involving computational environments, so that they can develop a computational literacy (diSessa, 2001).

Overall, the first goal of integration seems to be moderately met by studies, albeit with different assumptions, such as the transferability of unplugged learning we just discussed. However, since we established that all three goals should be met simultaneously, projects that are *only* providing early experiences in computing should strive to expand these opportunities for all students and to do so while advancing disciplinary learning.

Goal 2: Expanding computing education access.

One of the main arguments for introducing CT at the elementary level is that, by engaging all students at this age, students from typically underrepresented groups in science, engineering and computing (i.e., girls, students of color) will have **the opportunity** to become interested and proficient in computing before they develop their academic interests and self-select out of computational opportunities in middle school. However, in most of the studies we reviewed, determining the study's contribution to this goal was virtually impossible. In a third of the cases, the authors did not share the gender, racial, or socioeconomic distribution of participating elementary students. However, in all the studies we found, implementations were uniform across a classroom, with little discussion around engaging specific groups of students in computing—a one-size-fits-all approach.

While the absence of discussion on these topics at the elementary level does not constitute evidence that students from underrepresented groups are *not* being reached with CT initiatives, we believe researchers and educators should think purposefully about this goal of CT integration—and report their findings related to that goal. As Santo and colleagues (2020) conceptualized, issues of equity in computing can focus on *who* gets access to computing education, *how* CT is taught to meet the needs of all students, and *what* CT opportunities are provided to all students.

In terms of *who* gets access, we encourage teachers and researchers to think of which schools, classrooms, and groups of students get to participate in CT activities. In our work, we found that teachers were often integrating CT only as optional activities or "extension" opportunities for advanced students (Coenraad et al., 2020). Sometimes these decisions were based on the availability of resources (including time) and others due to classroom management concerns. However, no matter the reason, these types of implementations can lead to a self-selection process where only those with prior experience or interest in computing decide to participate—failing to contribute to the goal of exposing *all* students to CT.

As for *how* to teach CT to expand computing participation, we encourage researchers and teachers to think about the needs of their particular students, and design lessons and activities that build on the cultural competences of those students. In particular, the additional lens of culturally-responsive teaching (Ladson-Billings, 1995) can create CT opportunities where all students can participate and leverage their strengths to engage in computing. There is some beginning literature on this around computer science education (e.g., Eglash et al., 2013; Franklin et al., 2020; Searle & Kafai, 2015), but for CT integration in science and engineering, especially at the elementary level, this is lacking.

Finally, on the issue of *what* CT is taught, we refer to the issues of determining what constitutes an "early computing experience" discussed above. If we believe that there is an additional benefit to engaging with true computational environments (as opposed to replicating them in unplugged activities), then researchers

and educators need to understand how to make those environments accessible and usable for teachers and students. An approach where affluent schools are able to receive CT in the form of engaging with computational environments and underserved communities, lacking those same resources, only get unplugged approaches could create a disparity in the computing proficiency and confidence that students in each context develop. Unplugged activities should not be the default for those without the resources to do plugged CT activities. Similarly, we have seen that the highest levels of integration often require extensive material and researcher support. While these kinds of 'hothouse' implementations can provide insight into improving integration designs, they are less likely to be sustainable or scalable. It is important that we strive to find a sustainable way for *all* schools to receive the highest levels of integration—we should not conform to some schools receiving an Exist level while others (with more resources) are able to fully integrate CT at the Extend level.

Goal 3: Advancing scientific and engineering understanding

The contributions of the reviewed studies to this goal depended largely on their level of integration. In cases where CT was disjointed, science and engineering served only as a context for isolated CT activities. To be clear, it is possible that these implementations still advanced disciplinary understanding, but they did not do so *by engaging in CT*. In the second most common level of integration, studies at the Exist level showed a similar dynamic: students may have learned disciplinary content, but calling out existing practices and reframing as CT probably did little to further advance that understanding from the gains that typical instruction would provide.

In the Enhance level, albeit with the single example we found (Kalogiannakis et al., 2018), the scientific and engineering understanding students developed may not have been extended beyond what typical instruction would provide, but students were able to start making connections between that disciplinary content and computing principles. For example, using ScratchJr as an assessment tool where students create animations demonstrating their disciplinary learning can help students reformulate their understanding in a way that a computer can interpret. In this case, that representation constitutes a series of blocks that, put together, show an animation compatible with the student's understanding of a scientific process. The metacognitive processes involved in reformulating disciplinary understanding into computational representations could solidify that understanding (diSessa, 2000, 2018; Papert, 1980).

Lastly, studies at the Extend level were the most successful in advancing disciplinary understanding *through engagement in CT*. These implementations, which often required specialized tools like modeling and programming environments, allowed students to gain insights on content that could be less likely to emerge in traditional instruction. In fact, teachers from these implementations were often surprised at how much disciplinary content their students were able to grasp by engaging in CT within computational environments (Dickes et al., 2019). Learning units at this level took advantage of the affordances of CT and technologies to improve on learning over that provided in typical disciplinary instruction.

Research Agenda

As we reviewed the state of research in integrating CT into science and engineering, we now move to suggesting future research avenues from the unanswered questions that remain. The first recommendation stems from our difficulty in finding empirical studies of CT integration into science and engineering at the elementary level. Simply put, we need **more empirical evidence on how students can engage in CT-integrated science and engineering activities**. However, these studies should move beyond simply *describing* integration efforts, and in fact investigate **whether CT integration is, in fact, meeting the goals ascribed to it**.

While researchers and stakeholders argue that integrating CT can provide early experiences in computing, expand participation in computing education, and leverage computing for disciplinary learning, few—if any—projects have systematically evaluated those outcomes. As CT integration proliferates across K-5

education, it is imperative to assess *which* levels of integration are more likely to achieve these outcomes. For example, some researchers and educators adopt broad conceptualizations of CT that include practices already common in science and engineering education like data collection and analysis—embodying an Exist level of integration. But, the question remains:

> *Are CT integration efforts that simply call out CT in existing structures and curricula providing early computing experiences that facilitate developing proficiency in computing later, expanding computing participation, or leveraging computing for disciplinary learning?*

Furthermore, we need to understand how variability in integration is related to variability in meeting these goals. Are some subset of practices more applicable to meet some goals than others? How does taking an unplugged approach differ from using authentic computing environments in terms of developing computing proficiency? Are there science topics or curricular units that are most appropriate for CT integration that meets all three goals simultaneously?

Moreover, because we currently have studies that meet different goals individually, the field would greatly benefit from studies that show **how to support teachers and schools in integrating CT in ways that meet all three goals at the same time**. For example, how can teachers provide computing experiences that are authentic and foundational at the elementary level without compromising disciplinary learning or limiting those opportunities to a few students? How can teachers engage students from typically underrepresented groups or with little prior computational experiences in authentic computing experiences that advance their understanding of disciplinary content? Additionally, it would be informative to understand whether these goals are prioritized differently throughout the elementary years. As we saw in our analysis, some of the most advanced integrations were geared towards the older grades (4^{th} and 5^{th}). Are there ways to integrate CT while meeting all three goals in the lower grades? Are there authentic computing activities that can meet all three goals while maintaining developmental appropriateness? Research on Pre-K coding shows that very young students can understand basic computing principles (Bers, 2010; Sullivan & Bers, 2016), but the impact of these activities on equity of access to computing and development of disciplinary understanding is less clear.

The issue of developmental appropriateness is also reflected in the preparation of teachers and what they believe are adequate integrations of CT for students at each grade level. In one of our professional development sessions, we heard first grade teachers suggesting that their little learners would be limited to exploring CT using Fisher-Price's *code-a-pillar*, and that anything else would be beyond them. Several sessions later, these same first grade teachers were designing science lessons using ScratchJr! The literature clearly shows that all grades can engage with and learn from a wide array of CT skills and practices, but it would be important to understand how teacher *perceptions* of developmental appropriateness can impact the types of CT integrations that get implemented.

Relatedly, we also need more comprehensive and nuanced studies on **teacher education around CT integration**. Available research shows that teachers can learn about CT and integrate it into their lesson plans, but we have little evidence on the impact of that learning and lesson design on instructional practices (see Rich et al., 2020 for an exception). For example, how do different types of teacher learning (such as a CT module in a pre-service methods course and an in-service CT summer workshop) impact how teachers integrate CT in their classrooms? We have even less evidence on the impact that teacher learning around CT has on students' development of interest, confidence, and proficiency in CT. The field would also greatly benefit from investigating how teachers can develop the necessary skills and confidence to integrate CT to expand access to computing education. As we discussed above, even when teachers understand CT and integrate it in their classroom, the goal of making these opportunities available for *all* students may

take a secondary role if not addressed explicitly. How can we prepare teachers to create integration efforts that engage all their students?

Additionally, if we believe that CT integration under some conditions does in fact contribute to these goals and want to expand it, the field is lacking studies that **guide widespread adoption of integration**. Most studies reviewed above are projects where just a few teachers or classrooms participate, and are limited in their ability to inform how meso and macro-level factors impact integration. Related studies investigating the integration of CT into multiple disciplines at a school (e.g., Israel et al., 2015) and how districts enact computer science equity goals (Santo et al., 2020) suggest that school- and district-level structures and decisions greatly impact how teachers integrate CT and computing in their classrooms. Therefore, it is important that research in CT integration in science and engineering also investigates how districts and administrators can support teachers to integrate CT, and how factors beyond the instructional triangle (Ball & Forzani, 2007) can play a role in widespread integration.

Conclusions

In this paper, we have analyzed existing empirical literature to synthesize the fields' insights and gaps about integrating CT into science and engineering education. There are some key research trends to remark on. First, we found that the integration of CT into science and engineering is simply difficult to achieve: most studies added CT activities with no connection to disciplinary learning (Disjointed), changed no instructional practice but relabeled activities as CT (Exist), or employed a wealth of resources to create extensive and cohesive implementations (Extend). Clearly, balancing content and practices from two different disciplines while meeting curricular obligations is a tall order. However, the literature we reviewed also shows promise. These initial efforts can be built upon to reach higher levels of integration—the recognition of existing opportunities for CT integration is the first step in acting on those opportunities.

Secondly, we found some important challenges that seem to be exacerbated by the idiosyncrasies of elementary level education. For instance, the difficulty of merging two different disciplines (science or engineering and computing) is augmented by teachers' generalist focus, where few specialize on STEM or computing content in their training. This requirement of high expertise can result in teachers defaulting to only Disjointed or Exist levels of CT integration. This expertise challenge is, in turn, aggravated by teacher educators and PD designers who, often lacking the same expertise in computing, model integration at Exist or Disjointed levels. Additionally, efforts to make computing—which in professional contexts can be extremely complex and require rich technical knowledge—developmentally appropriate for 5-11 year-old children can blur the lines between scientific inquiry, engineering practices, and CT. In the studies we reviewed, it is unclear whether all of the CT operationalizations we found would, in fact, (a) contribute to the development of proficiencies that would help students manipulate computational environments and devices in the future and (b) advance disciplinary understanding.

Finally, the literature shows that, while many argue that integrating CT in the early grades can be a tool for expanding computing education, studies focused on science and engineering contexts have yet to document progress towards that substantial goal. It is possible that the field is still resolving *what* CT can be integrated into science and engineering and whether these integrations can promote disciplinary understanding. But, this focus on CT definitions and curricular overlaps is neglecting a critical examination of *who* gets to participate in these initiatives. The challenge for the field is to investigate how to meet all three goals at the same time: integrate CT to develop confidence and proficiency in computing, expand computing education, and advance disciplinary learning.

References

Association for Computing Machinery. (2016). *K-12 Computer Science Framework*. www.k12cs.org

Bain, C., Dabholkar, S., & Wilensky, U. (2020). Confronting Frame Alignment in CT Infused STEM Classrooms. In S. C. Kong, H. U. Hoppe, T. C. Hsu, R. H. Huang, B. C. Kuo, K. Y. Li, C. K. Looi, M. Milrad, J. L. Shih, K. F. Sin, K. S. Song, M. Specht, F. Sullivan, & J. Vahrenhold (Eds.), *Proceedings of International Conference on Computational Thinking Education 2020*. The Education University of Hong Kong.

Ball, L., & Forzani, F. M. (2007). What Makes Education Research "Educational"? *Educational Researcher*, *36*(9), 529–540. https://doi.org/10.3102/0013189X073

Barr, V., & Stephenson, C. (2011). Bringing computational thinking to K-12: what is Involved and what is the role of the computer science education community? *ACM Inroads*, *2*(1), 48–54. https://doi.org/10.1145/1929887.1929905

Basu, S., Biswas, G., Kinnebrew, J., & Rafi, T. (2015). Relations between modeling behavior and learning in a Computational Thinking based science learning environment. *Proceedings of the 23rd International Conference on Computers in Education, ICCE 2015*, 184–189.

Basu, S., Biswas, G., Sengupta, P., Dickes, A., Kinnebrew, J. S., & Clark, D. (2016). Identifying middle school students' challenges in computational thinking-based science learning. *Research and Practice in Technology Enhanced Learning*, *11*(1), 13. https://doi.org/10.1186/s41039-016-0036-2

Bers, M. U. (2010). The TangibleK robotics program: Applied computational thinking for young children. *Early Childhood Research and Practice*, *12*(2), 1–20.

Cabrera, L., Coenraad, M., Killen, H., Ketelhut, D. J., & Plane, J. (n.d.). Integrating Computational Thinking into Elementary Science Teaching in a Pre-Service Course. *In Preparation*.

Cabrera, L., Coenraad, M., Mills, K. M., McGinnis, J. R., & Ketelhut, D. J. (2020). Preservice Teachers' Self-Efficacy and Computational Thinking: A Mixed-Methods Approach. *American Educational Research Association Conference 2020 (AERA)*.

Cabrera, L., McGinnis, J. R., Ketelhut, D. J., Hestness, E. E., Mills, K. M., & Jeong, H. (2019). Preservice Teachers' Changes in Self-Efficacy Regarding Computational Thinking. *NARST 92nd Annual International Conference*.

Coenraad, M., Mills, K. M., Byrne, V., & Ketelhut, D. J. (2020). Supporting Teachers to Integrate Computational Thinking Equitably. *Proceedings of Research on Equity and Sustained Participation in Engineering, Computing, and Technology (RESPECT 2020)*.

Computer Science Teachers Association, & International Society for Technology in Education. (2011). *Computational Thinking: Leadership Toolkit*. http://www.iste.org/docs/ct-documents/ct-leadershipt-toolkit.pdf

Dickes, A. C., Kamarainen, A., Metcalf, S. J., Gün-Yildiz, S., Brennan, K., Grotzer, T., & Dede, C. (2019). Scaffolding ecosystems science practice by blending immersive environments and computational modeling. *British Journal of Educational Technology*, *50*(5), 2181–2202. https://doi.org/10.1111/bjet.12806

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

diSessa, A. A. (2018). Computational Literacy and "The Big Picture" Concerning Computers in Mathematics Education. *Mathematical Thinking and Learning*, *20*(1), 3–31. https://doi.org/10.1080/10986065.2018.1403544

Dwyer, H. A., Boe, B., Hill, C., Franklin, D., & Harlow, D. (2014). Computational Thinking for Physics: Programming Models of Physics Phenomenon in Elementary School. *2013 Physics Education Research Conference Proceedings*, *February 2014*, 133–136. https://doi.org/10.1119/perc.2013.pr.021

Eglash, R., Gilbert, J. E., Taylor, V., & Geier, S. R. (2013). Culturally Responsive Computing in Urban, After-School Contexts: Two Approaches. *Urban Education*, *48*(5), 629–656. https://doi.org/10.1177/0042085913499211

Ehsan, H., Rehmat, A. P., & Cardella, M. E. (2020). Computational thinking embedded in engineering design: capturing computational thinking of children in an informal engineering design activity. *International Journal of Technology and Design Education*, *0123456789*. https://doi.org/10.1007/s10798-020-09562-5

Franklin, D., Weintrop, D., Palmer, J., Coenraad, M., Cobian, M., Beck, K., Rasmussen, A., Krause, S., White, M., Anaya, M., & Crensh, Z. (2020). Scratch encore: The design and pilot of a culturally-relevant intermediate scratch curriculum. *Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE*, 794–800. https://doi.org/10.1145/3328778.3366912

Garvin, M., Killen, H., Plane, J., & Weintrop, D. (2019). Primary School Teachers' Conceptions of Computational Thinking. *SIGCSE '19 Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, *2*, 899–905. https://doi.org/10.1145/3287324.3287376

Grover, S. (2011). Robotics and engineering for middle and high school students to develop computational thinking. *Annual Meeting of the American Educational Research Association*, *650*, 1–15. http://web.stanford.edu/~shuchig/docs/AERA2011-Shuchi Grover-Robotics and Engineering for Middle and High School Students to Develop Computational Thinking.pdf

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

Gürbüz, H., Evlioğlu, B., Erol, Ç. S., Gülseçen, H., & Gülseçen, S. (2017). "What's the Weather Like Today?": A computer game to develop algorithmic thinking and problem solving skills of primary school pupils. *Education and Information Technologies*, *22*(3), 1133–1147. https://doi.org/10.1007/s10639-016-9478-9

Hestness, E., Ketelhut, D. J., Plane, J., Razler, B., & McGinnis, R. J. (2018). Computational thinking professional development for elementary science educators: Examining the design process. *The Society for Information Technology and Teacher Education (SITE)*.

Hestness, E., Mills, K. M., McGinnis, J. R., Ketelhut, D. J., Jeong, H., & Cabrera, L. (2019). Preservice Teachers' Beliefs About CT Integration in Elementary Science Instruction. *NARST 92nd Annual International Conference*.

Horizon Research. (2019). *Highlights from the 2018 NSSME+*.

Horn, M. S., Brady, C., Hjorth, A., Wagh, A., & Wilensky, U. (2014). Frog Pond: A code-first learning environment on evolution and natural selection. *ACM International Conference Proceeding Series*, 357–360. https://doi.org/10.1145/2593968.2610491

Hynes, M. M., Moore, T. J., Cardella, M. E., Tank, K. M., Purzer, S., Menekse, M., & Brophy, S. P. (2016). Inspiring computational thinking in young children's engineering design activities (Fundamental). *ASEE Annual Conference and Exposition, Conference Proceedings*, *2016-June*. https://doi.org/10.18260/p.25732

Israel, M., & Lash, T. (2019). From classroom lessons to exploratory learning progressions: mathematics + computational thinking. *Interactive Learning Environments*. https://doi.org/10.1080/10494820.2019.1674879

Israel, M., Pearson, J. N., Tapia, T., Wherfel, Q. M., & Reese, G. (2015). Supporting all learners in school-wide computational thinking: A cross-case qualitative analysis. *Computers and Education*, *82*, 263–279. https://doi.org/10.1016/j.compedu.2014.11.022

Jaipal-Jamani, K., & Angeli, C. (2017). Effect of Robotics on Elementary Preservice Teachers' Self-Efficacy, Science Learning, and Computational Thinking. *Journal of Science Education and Technology*, *26*(2), 175–192. https://doi.org/10.1007/s10956-016-9663-z

Jaipal-Jamani, K., & Angeli, C. (2018). Developing Teacher Self-Efficacy to Teach Science and Computational Thinking with Educational Robotics: Using Scaffolded Programming Scripts. In C. Hodges (Ed.), *Self-Efficacy in Instructional Technology Contexts* (pp. 183–203). https://doi.org/10.1007/978-3-319-99858-9_11

Kalogiannakis, M., Ampartzaki, M., Papadakis, S., & Evangelia, S. (2018). Teaching natural science concepts to young children with mobile devices and hands-on activities. A case study. *International Journal of Teaching and Case Studies*, *9*(2), 171. https://doi.org/10.1504/ijtcs.2018.10011893

Kaya, E., Yesilyurt, E., Newley, A., & Deniz, H. (2019). Examining the Impact of a Computational Thinking Intervention on Pre-Service Elementary Science Teachers' Computational Thinking Teaching Efficacy Beliefs, Interest and Confidence. *Journal of Computers in Mathematics and Science Teaching*, *38*(4), 385–392.

Ketelhut, D. J., Cabrera, L., McGinnis, R. J., Plane, J., Coenraad, M., Killen, H., & Mills, K. M. (2019). Exploring the Integration of computational Thinking into Preservice Elementary Science Teacher Education. *National Science Foundation STEM+C PI Meeting*. http://stemcsummit.edc.org/slides/DianeJass.pdf

Ketelhut, D. J., Hestness, E., Cabrera, L., Jeong, H., Plane, J., & McGinnis, J. R. (2019). Examining the Role of Mentor Teacher Support in a Professional Learning Experience for Preservice Teachers on Integrating Computational Thinking into Elementary Science Education. In K. Graziano (Ed.), *Proceedings of Society for Information Technology & Teacher Education International Conference 2019* (pp. 2281–2285). Association for the Advancement of Computing in Education (AACE). https://www.learntechlib.org/p/207966

Ketelhut, D. J., Mills, K., Hestness, E., Cabrera, L., Plane, J., & McGinnis, J. R. (2019). Teacher Change Following a Professional Development Experience in Integrating Computational Thinking into Elementary Science. *Journal of Science Education and Technology*, *Computational Thinking from a Disciplinary Perspective*, 1–15. https://doi.org/10.1007/s10956-019-09798-4

Killen, H., Coenraad, M., Byrne, V. L., Cabrera, L., & Ketelhut, D. J. (2020). Reimagining Computational Thinking Professional Development: Benefits of a Community of Practice Model. *Proceedings of the International Conference of the Learning Sciences (ICLS 2020)*, 2125–2132.

Ladson-Billings, G. (1995). Toward a Theory of Culturally Relevant Pedagogy. *American Educational Research Journal*. https://doi.org/10.3102/00028312032003465

Lee, I., Grover, S., Martin, F., Pillai, S., & Malyn-Smith, J. (2020). Computational Thinking from a Disciplinary Perspective: Integrating Computational Thinking in K-12 Science, Technology, Engineering, and Mathematics Education. *Journal of Science Education and Technology*, *29*(1), 1–8. https://doi.org/10.1007/s10956-019-09803-w

Lee, I., & Malyn-Smith, J. (2020). Computational Thinking Integration Patterns Along the Framework Defining Computational Thinking from a Disciplinary Perspective. *Journal of Science Education and Technology*, *29*(1), 9–18. https://doi.org/10.1007/s10956-019-09802-x

Leonard, A. E., Dsouza, N., Babu, S. V., Daily, S. B., Jörg, S., Waddell, C., Parmar, D., Gundersen, K., Gestring, J., & Boggs, K. (2015). Embodying and Programming a "Constellation" of Multimodal Literacy Practices: Computational Thinking, Creative Movement, Biology, & Virtual Environment Interactions. *Journal of Language and Literacy Education*, *11*(2), 64–93.

Luo, F., Antonenko, P. D., & Davis, E. C. (2020). Exploring the evolution of two girls' conceptions and practices in computational thinking in science. *Computers and Education*, *146*(May 2019), 103759. https://doi.org/10.1016/j.compedu.2019.103759

Malyn-Smith, J., Blustein, D., Pillai, S., Parker, C. E., Gutowski, E., & Diamonti, A. J. (2017). *Building the Foundational Skills Needed for Success in Work at the Human-Technology Frontier*.

McGinnis, J. R., Hestness, E., Mills, K., Ketelhut, D. J., Cabrera, L., & Jeong, H. (2020). Preservice Science Teachers' Beliefs about Computational Thinking Following a Curricular Module within an Elementary Science Methods Course. *Contemporary Issues in Technology and Teacher Education*, *20*(1), 85–107.

Mensan, T., Osman, K., & Majid, N. A. A. (2020). *Development and Validation of Unplugged Activity of Computational Thinking in Science Module to Integrate Computational Thinking in Primary Science Education*. *31*(2), 142–149.

Mouza, C., Yang, H., Pan, Y.-C., Yilmaz Ozden, S., & Pollock, L. (2017). Resetting educational technology coursework for pre-service teachers: A computational thinking approach to the development of technological pedagogical content knowledge (TPACK). *Australasian Journal of Educational Technology*, *33*(3), 61–76. https://doi.org/10.14742/ajet.3521

National Research Council. (2011). *Report of a Workshop on the Pedagogical Aspects of Computational Thinking*. https://doi.org/10.17226/13170

National Science and Technology Council. (2018). *Charting a Course for Success: America's Strategy for STEM Education* (Issue December).

Papert, S. (1980). Mindstorms: Computers, children, and powerful ideas. *NY: Basic Books*.

Pinto-Llorente, A. M., Casillas-Martín, S., Cabezas-González, M., & García-Peñalvo, F. J. (2018).

Building, coding and programming 3D models via a visual programming environment. *Quality and Quantity*, *52*(6), 2455–2468. https://doi.org/10.1007/s11135-017-0509-4

Rich, K. M., Yadav, A., & Larimore, R. A. (2020). Teacher implementation profiles for integrating computational thinking into elementary mathematics and science instruction. *Education and Information Technologies*, *25*(4), 3161–3188. https://doi.org/10.1007/s10639-020-10115-5

Ryoo, J. J. (2019). Pedagogy that supports computer science for All. *ACM Transactions on Computing Education*, *19*(4). https://doi.org/10.1145/3322210

Sands, P., Yadav, A., & Good, J. (2018). Computational thinking in K-12: In-service teacher perceptions of computational thinking. *Computational Thinking in the STEM Disciplines: Foundations and Research Highlights*, 151–164. https://doi.org/10.1007/978-3-319-93566-9_8

Santo, R., Delyser, L. A., Ahn, J., Pellicone, A., Aguiar, J., & Wortel-London, S. (2019). Equity in the Who, How and What of Computer Science Education: K12 School District Conceptualizations of Equity in "CS for All" Initiatives. *Proceedings of the 2019 Research on Equity and Sustained Participation in Engineering, Computing, and Technology, RESPECT 2019*. https://doi.org/10.1109/RESPECT46404.2019.8985901

Searle, K. A., & Kafai, Y. B. (2015). Boys' needlework: Understanding gendered and indigenous perspectives on computing and crafting with electronic textiles. *ICER 2015 - Proceedings of the 2015 ACM Conference on International Computing Education Research*, 31–40. https://doi.org/10.1145/2787622.2787724

Sengupta, P., & Farris, A. V. (2012). Learning kinematics in elementary grades using agent-based computational modeling: A visual programming-based approach. *ACM International Conference Proceeding Series*, 78–87. https://doi.org/10.1145/2307096.2307106

Sengupta, P., Kinnebrew, J. S., Basu, S., Biswas, G., & Clark, D. (2013). Integrating computational thinking with K-12 science education using agent-based computation: A theoretical framework. *Education and Information Technologies*, *18*(2), 351–380. https://doi.org/10.1007/s10639-012-9240-x

Shulman, L. (1986). Those Who Understand: Knowledge Growth in Teaching. *Educational Researcher*, *15*(2), 4–14. http://www.dnv.com

Sullivan, A., & Bers, M. U. (2016). Robotics in the early childhood classroom: learning outcomes from an 8-week robotics curriculum in pre-kindergarten through second grade. *International Journal of Technology and Design Education*, *26*(1), 3–20. https://doi.org/10.1007/s10798-015-9304-5

Tedre, M., & Denning, P. J. (2016). The long quest for computational thinking. *Proceedings of the 16th Koli Calling International Conference on Computing Education Research - Koli Calling '16*, 120–129. https://doi.org/10.1145/2999541.2999542

Toma, R. B., Lederman, N. G., Jiménez, J., & A., M. V. J. (2019). Exploring students' acceptance of coding activities during integrative STEM lessons. *NARST 2019 Annual International Conference*, 1–5.

Waterman, K. P., Goldsmith, L., & Pasquale, M. (2019). Integrating Computational Thinking into Elementary Science Curriculum: an Examination of Activities that Support Students' Computational

Thinking in the Service of Disciplinary Learning. *Journal of Science Education and Technology*, 53–64. https://doi.org/10.1007/s10956-019-09801-y

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

Wing, J. M. (2006). Computational Thinking. *Communications of the ACM*. https://doi.org/10.1145/1227504.1227378

Wing, J. M. (2010). Research notebook: Computational thinking—What and why? *The Link Magazine*, *Spring*, 1–6.

Yadav, A., Krist, C., Good, J., & Nadire Caeli, E. (2018). Computational thinking in elementary classrooms: measuring teacher understanding of computational ideas for teaching science. *Computer Science Education*. https://doi.org/10.1080/08993408.2018.1560550

Yadav, A., Mayfield, C., Zhou, N., Hambrusch, S., & Korb, J. T. (2014). Computational Thinking in Elementary and Secondary Teacher Education. *ACM Transactions on Computing Education*, *14*(1), 1–16. https://doi.org/10.1145/2576872