The Integration of Computational Thinking in Early Childhood and Elementary Education

Tamara J. Moore, Professor of Engineering Education, Purdue University
Anne T. Ottenbreit-Leftwich, Associate Professor of Instructional Systems Technology, Indiana University

What is CT/CS?

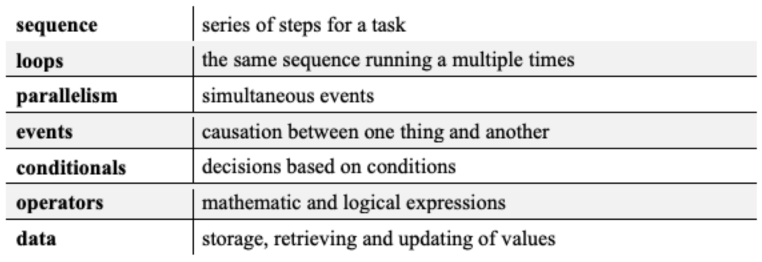
CT Definitions and Classifications

Commonly considered the article that began the current mainstream focus on computational thinking, Wing (2006) indicated that computational thinking includes the following characteristics: (1) conceptualizing, not programming, (2) fundamental, not rote skill, (3) a way that humans, not computers, think, (4) complements and combines mathematical and engineering thinking, (5) ideas, not artifacts, and (6) for everyone, everywhere. one of the seminal authors around computational thinking. In fact, Wing (2006), aggressively recommended that computational thinking needs to be taught to every student as a way to solve problems and potentially use computer science logic to solve a myriad of everyday problems. In 2008, Wing (2008) defined CT as "taking an approach to solving problems, designing systems and understanding human behaviour that draws on concepts fundamental to computing'' (p. 3717). In general, CT definitions typically seem to encompass the process of identifying a problem and creating potential solutions so that a computer (whether that be a human or machine) could potentially implement that solution.

Gretter and Yadav (2016) referred to computational thinking as the ability to "think like a computer scientist" (p. 511) which utilizes specific problem solving skills focused around algorithmic thinking, pattern recognition, abstraction, and decomposition. Yadav et al. (2016) described computational thinking (CT) as "breaking down complex problems into more familiar/manageable sub-problems (problem decomposition), using a sequence of steps (algorithms) to solve problems, reviewing how the solution transfers to similar problems (abstraction), and finally determining if a computer can help more efficiently solve those problems (automation)" (p. 565). In fact, a new advanced placement course was developed to specifically focus on these capabilities, as opposed to pure computer science, called Advanced Placement Computer Science Principles (Gretter & Yadav, 2016). The AP CSP course "focuses on developing students' 21st century skills, such as analyzing and representing data, understanding how the Internet functions, and grasping how computing impacts people and society" (Gretter & Yadav, 2016, p. 511).

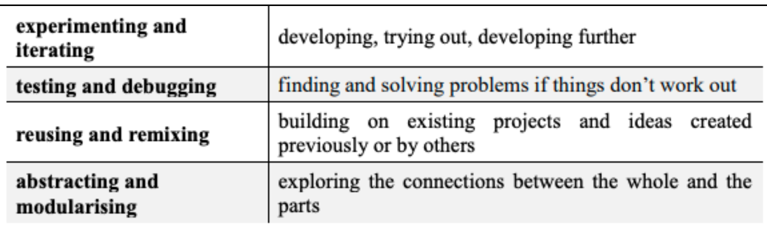
Many have suggested how to discuss and conceptualize computational thinking. Perhaps the most popular CT framework comes from Brennan and Resnick (2012). They established their

CT framework based on their investigations of students' Scratch projects. They divided CT into *concepts*, *practices*, and *perspectives* (see Figures 1-3 below).



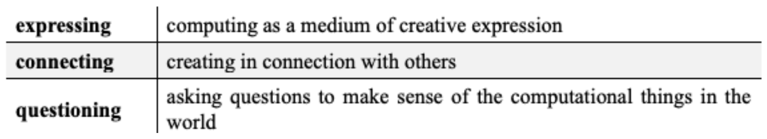
sequence	series of steps for a task
loops	the same sequence running a multiple times
parallelism	simultaneous events
events	causation between one thing and another
conditionals	decisions based on conditions
operators	mathematic and logical expressions
data	storage, retrieving and updating of values

Figure 1. CT Concepts.



experimenting and iterating	developing, trying out, developing further
testing and debugging	finding and solving problems if things don't work out
reusing and remixing	building on existing projects and ideas created previously or by others
abstracting and modularising	exploring the connections between the whole and the parts

Figure 2. CT Practices.



expressing	computing as a medium of creative expression
connecting	creating in connection with others
questioning	asking questions to make sense of the computational things in the world

Figure 3. CT Perspectives.

Policy and State Standards

Global CT Policies

Computational thinking has become an increased interest in K-12 education across the globe. Balanskat and Engelhardt (2014) surveyed 17 European countries to identify how they attempted to incorporate CT into the K-12 curriculum. The UK implemented CT into courses across disciplines (including CS, information technology, and digital citizenship)(Brown, Sentance, Crick, & Humphreys, 2014). In Australia, a CT course was incorporated into their primary and secondary school curriculum (Falkner, Vivian, & Falkner, 2014). Poland implemented a three-stage process for integrating CT courses in their primary and secondary schools. The final stage required computer science in their high school final examinations (Sysło & Kwiatkowska, 2015). South Korea has also incorporated more than 34 hours of computer instruction in each grade K-12. They adopted a national curriculum and textbook around computer science (Heintz et al., 2016).

US CT Policies

In a review of the CS standards policies for all 50 states, Guo and Ottenbreit-Leftwich (2020) found that 34 states had published computer science standards on their websites as of January 2020 and one state, Montana, announced on the website that the state plans to publish the CS standard in 2020 fall semester. However, the US has no universal CS standard accepted by all states in the U.S., and states designed CS standards for their own sake to coordinate the whole curriculum. In addition, Guo and Ottenbreit-Leftwich (2020) found that 22 out of 34 states adopted the Computer Science Teacher Association (CSTA) CS framework, and 12 states created CS related standards independently. Sometimes CT/CS standards were their own separate subject area (n=11), and sometimes CT/CS was included into broader computing standards with digital literacy and citizenship (n=13). The CT/CS standards were also placed in different areas, such as Indiana, where CS/CT standards were housed within the science standards subject area (Guo & Ottenbreit-Leftwich, 2020).

The Computer Science framework from k12cs.org was published in 2017 and helped guide the development of many of the CS standards (*K–12 Computer Science Framework*, 2016) The framework identified five core concepts for K-12 Computer Science: (1) Computing Systems, (2) Networks and the Internet, (3) Data and Analysis, (4) Algorithms and Programming, and the (5) Impact of Computing. Following this, the Computer Science Teacher Association (CSTA) established Computer Science standards that were the most commonly used K-12 CS standards with 22 out of the 34 states directly adopting these standards. Of the 12 states that did not directly use the national CSTA standards, seven states explicitly mentioned and included elements of computational thinking in their standards. Five states incorporated ideas around computational thinking from ISTE standards, which includes one standard called "to be the computational thinker." However, other states separated Computational Thinking as an individual concept. For example, Colorado and Massachusetts organized Computational Thinking as an important concept in the computer science standards. Arkansas connected Computational Thinking with Problem Solving in their state-created Computer Science Standards (Guo & Ottenbreit-Leftwich, 2020). Guo and Ottenbreit-Leftwich (2020) also identified that 32 out of the 34 states had CS curriculum standards beginning at the kindergarten level.

CT/CS Integration into Subject Area Standards

Computational thinking is described differently across standards and frameworks. For example, the CS framework did not include computational thinking as a concept, but described computational thinking as the practices students use in computer science. In some states, they integrated computational thinking into other subject areas. For example, New Jersey's standards featured computational thinking as a key concept of Engineering and Design. In another example, Michigan adopted the Next Generation Science Standard (NGSS) and computational thinking was marked as one of the core concepts in the Science and Engineering Practices dimension. It was common for states to reference teaching computational thinking with other disciplines (e.g., science, math, engineering) due to the sharing concepts among diverse disciplines (Guo & Ottenbreit-Leftwich, 2020).

Rationale for CT: 21st Century Skills and Digital Citizenship

Scholars have also expressed the difficulties associated with separating CT/CS concepts from other computing-related areas, such as computer literacy, information technology, educational technology, digital citizenship, and computational thinking (Mouza et al., 2018). In addition, scholars have emphasized that we need to incorporate more concepts related to CS, such as computational thinking (Wing, 2008) and digital citizenship, as an essential skill of participating in society (Mossberger et al., 2007).

Moreover, some suggested that the goal of education is to produce strong and contributing citizens (e.g., Roosevelt, 2008). However, with the fast development of technology, especially computers and the Internet, the concept of citizenship has expanded to incorporate the digital world. Scholars have claimed that teaching K-12 digital citizenship will help students become citizens that will thrive and contribute to our digital society (Ribble, 2015), as well as provide equal opportunities for all students in the new digital society (Vogel et al., 2017). Gretter and Yadav (2016) also pointed out that to be prepared for today's participatory culture, students need 21st century skills that can enable them to be creators, as opposed to passive receivers of information. As our society is increasingly shifting towards digital engagement, students need to develop the ability to deconstruct problems and solve them utilizing the power of computers. This requires computational thinking skills. Scholars have argued that computational thinking skills require students to develop both domain-specific and general problem-solving skills (Yadav, Good, et al., 2017). Yadav, Good, et al. (2017) described the importance of incorporating CT into compulsory education:

> "Computational thinking is a broadly applicable competence domain, which is important for individuals to be successful in today's technological society…Given that computational thinking has been highlighted as an ubiquitous twenty-first century skill and the emphasis placed on the need to embed CT in primary and secondary schooling, we need to focus on better understanding how computational thinking tools support learners" (p. 1064).

PreK-5 Strategies for CT Learning

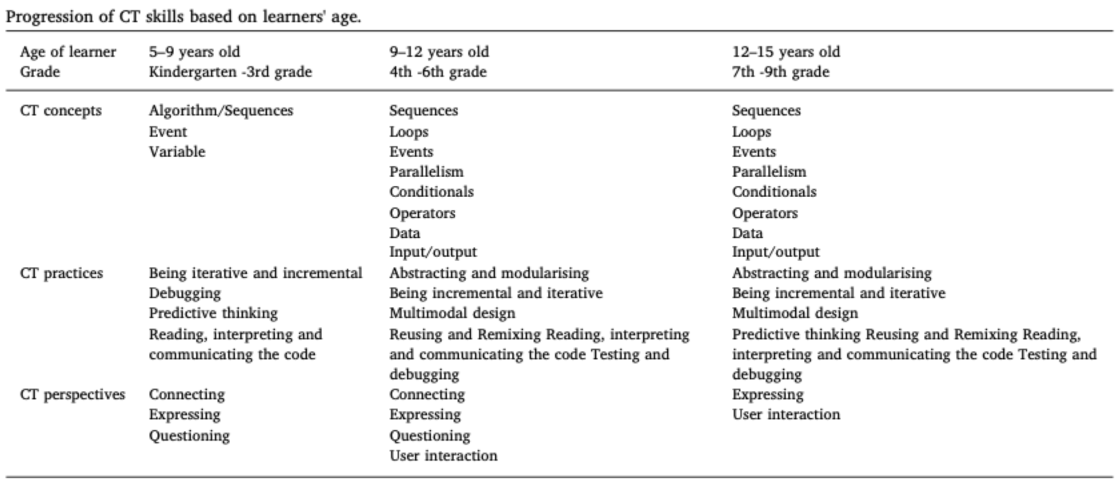
This section provides a look into three areas that need consideration when addressing CT learning with preK-5 students. In the following subsections, we will provide overviews of developmental appropriateness for CT learning strategies, using multiple representations in CT learning, and making CT learning active, hands-on, and minds-on.

Developmental Appropriateness for CT Learning Strategies

Teaching CT to children has been shown to be a good tool for mind development (Buitrago Flórez et al., 2017). As Bruner (1960) stated, "any subject can be taught effectively in some intellectually honest form to any child at any stage of development" (p. 33). Conceptual development in CT will be initially localized in the learning task in which the concept is learned, then through multiple instances of addressing a concept or skill, students will begin to develop deeper, more abstract ideas regarding the concepts. CT skills invoke modeling. In regards to modeling tasks, Lesh and Harel (2003) state, "if we examine a student's performances across a series of related activities, it is clear that his or her apparent stage of development often varies considerably across tasks" (p. 186). Therefore, this would also be true of complex CT tasks that allow for students to express their CT models in ways that allow for refinement and testing. Therefore, it is important to consider how to be able to break down a complex topic like computer science and CT into manageable parts for younger students.

Based on the Brennan and Resnick (2012) framework, Zhang and Nouri (2019) examined all Scratch-based empirical studies with Kindergarten through ninth-grade students. After reviewing 55 studies, they developed a progression of CT skills (concepts, practices, and perspectives) based on students' ages (see Figure 4 below). This progression, while not yet well studied as a final product for effectiveness, is a good start to consider which CT skills to emphasize for different age levels.

Lee and Mayn-Smith (2020) investigated CT learning progressions by examining funded projects at NSF-funded workshops. In K-2, they identified that abstraction (in which they included patterns and representation) included looking for patterns in works, representing people with glyphs, and representing shapes and movements; algorithms included instructing Bee-Bot (see Robotics/Devices Section below) through a maze and instructing humans as if they were robots; programming and development included what they termed "everyday mechanisms" such as money exchange and guess my number examples; data collection and analysis included sorting objects and using tally marks for counting; and finally, modeling and simulation included running an experiment and comparing solutions. In grades 3-5, they found that abstraction included making abstract art and storyboarding; algorithms included programming robots or developing instructions for Lego builds; programming and development included CAD, animation of clock hands, and Scratch animations; data collection and analysis included comparing solutions, guessing the rules, design tasks, and simulation to produce data; and finally, modeling and simulation included developing models – mathematical models, amusement park rides, invasive species, and ecosystems (Lee & Malyn-Smith, 2020).


Progression of CT skills based on learners' age.

Age of learner Grade	5–9 years old Kindergarten -3rd grade	9–12 years old 4th -6th grade	12–15 years old 7th -9th grade
CT concepts	Algorithm/Sequences Event Variable	Sequences Loops Events Parallelism Conditionals Operators Data Input/output	Sequences Loops Events Parallelism Conditionals Operators Data Input/output
CT practices	Being iterative and incremental Debugging Predictive thinking Reading, interpreting and communicating the code	Abstracting and modularising Being incremental and iterative Multimodal design Reusing and Remixing Reading, interpreting and communicating the code Testing and debugging	Abstracting and modularising Being incremental and iterative Multimodal design Predictive thinking Reusing and Remixing Reading, interpreting and communicating the code Testing and debugging
CT perspectives	Connecting Expressing Questioning	Connecting Expressing Questioning User interaction	Expressing User interaction

Figure 4. Progression of CT skills based on learners' age (Zhang & Nouri, 2019), p. 19.

Students at any age will have inherent challenges if they are new to CT. These challenges can include things such as getting to know the intricacies of programming languages or platforms, learning how to "talk" to a machine with instructions the machine can understand, and getting unstuck when their program is not working (correctly or at all). If we add in the learning development of elementary age students, these challenges can become increasingly difficult. Researchers have started to study and build these CT learning trajectories for early grades. Rich et al. (2018) worked to develop a decomposition learning trajectory for K-8, showing such dimensions as “code is reusable” and “code can be written in small parts.” This type of work needs to continue and be developed with greater detail for all of the CT skills. Therefore, as CT learning is a fairly new area to research with students at this age, pulling from the literature on learning progressions/trajectories for elementary students, particularly in mathematics and literacy, may help with addressing some of the challenges that students will face as they begin to develop CT skills.

Using Multiple Representations in CT Learning

The purpose of teaching computational thinking is ultimately geared toward machine automation (Yadav, Goode et al., 2017). The ideas behind coding a machine to perform certain tasks require very abstract concepts. As stated in the developmentally appropriate strategies section, students in early grades need to start with more concrete ideas then progress toward more abstract ones. As students are developing ideas around computational thinking, helping students make the transitions among coding concepts is key. Multiple representations are a good way to do this.

Generally, coding in secondary and postsecondary classrooms involve multiple representations of code. These multiple representations can include real life examples, everyday language, pseudocode, flowcharts, code tracing/tracking charts and tables, coding languages, etc. (Malik et al., 2019). Most of these representations of algorithms and debugging within coding are much too abstract for children to begin with (Fessakis et al., 2013). Therefore, it is important that students, especially K-5 students, are introduced to computational concepts using multiple representations starting with more concrete ideas and then progressing to more abstract. However, during this learning process, students need to be making translations between and among these representations (Moore, Brophy et al., 2020). As students are developing computational concepts, they will use concrete, pictorial, motor, language, and symbolic representations (Bers, 2018b; Lesh & Doerr, 2003; Nathan et al., 2013). These representations should be developmentally appropriate approximations of the processes of coding. For example, Moore, Brophy et al. (2020) set up purposeful analogous representations for processes of coding to test the developmental appropriateness of these representations and the translation between them in a study of second-grade students using a computing device (see Robot Mouse in Robotics/Devices Section below). In this study, directional coding cards represented flow charts of coding and these cards could also be used for code tracing and debugging, coding with directional buttons were a representation of coding language, and students used language and gesture as pseudocode for planning. For second-grade students, translation between these representations often involved high cognitive demand. Therefore, students were seen developing intermediary representations to help manage their cognitive load and make the needed translations between representations to accomplish the CT tasks being asked of them (Moore, Brophy et al., 2020).

But as students develop, it is important to use concrete manipulatives purposefully and with care. Aggarwal et al. (2017) compared 3rd-5th grade students using physical manipulatives to develop code within the Kodu curriculum to students who did not use the manipulatives. The students were divided into groups that used flashcards and tiles before entering code into the Kodu Game Lab or just paper and pencil representations before entering code into the Kodu Game Lab. Here the two groups overall performed similarly content-wise, but the group with the manipulatives did better on tasks that involved syntax but this was time consuming while the group without the manipulatives did better at completing the tasks in a timely, iterative manner getting immediate feedback from the software. The authors suggest that manipulatives may have diminishing returns and therefore should be scaffolded carefully (Aggarwal et al., 2017). These studies shed light on how representations can be used to develop CT competencies and how students learn with multiple representations. However, much more research is needed to understand which practices are most effective in helping students progress towards high-quality CT learning.

Making CT Learning Active, Hands-on, and Minds-on

CT is more than just coding. CT offers broad opportunities for students to engage in physical manipulations, movement, and motor skills (Bers, 2018b; Byers & Walker, 1995). CT learning also requires students to solve problems algorithmically and develop technological fluency and language (Bers, 2010; Papert, 1980). Students must learn the language of coding through "learning about" it as well as "using" it (Bers, 2018a). These ideas suggest that students need to engage with the CT content in multiple hands-on and minds-on ways.

Curricular innovations such as those described by Aggarwal et al. (2017) above that involve physical manipulatives provide an entry point for learners. Other curricular innovations involve whole body movement and large-scale concrete manipulation of objects that encourage CT. The Puppy Playground (Ehsan et al., 2019) engages children in engineering design to develop a play space for a dog using "Big Blue Blocks." This activity could be implemented in any learning environment that can handle the size of the blocks, but also could be scaled using smaller blocks. In small groups, students worked together for a client (a kindergarten student who wants to let her puppy play in the yard). The criteria for the design was to keep the puppy from escaping, allow for play and exercise, and make the playground aesthetically pleasing by including patterns in the design. The physical nature of this task with the larger blocks allows students to think through the problem from the end-user's perspective - that of the puppy. The studies of this task show the students participating in the CT concepts of problem decomposition, pattern recognition, debugging, algorithm and procedure, simulation, and abstraction (Ehsan et al., 2019; 2020). But the active, hands-on nature of the task helps students with connection between very characteristic engineering concepts such as user-centered design and the CT principles the activity was designed to elicit.

In a more complex museum or science center exhibit, *Computing for the Critters*, was also designed to integrate engineering and CT through the context of designing an automated way to deliver medicine to all of the animals in a veterinary hospital. The exhibit has five interconnected sections: (1) a place to learn about CT and the context of the rest of the exhibit, (2) a physical maze for children to climb and act out the scenario, (3) a station to plan and test routes through the maze, (4) panels with details of different types of engineering, and (5) an interactive coding video game (Ehsan, Ohland et al., 2018). The CT in this exhibit is elicited as the students interacted with the different exhibit elements. For example, problem decomposition, patterns, parallelization, and simulation were elicited in the physical maze when children worked to physically get the medicine to all of the animals, whereas, algorithms and procedures, debugging, were elicited during the interactive coding game (Fagundes et al., 2020). These examples represent a variety of different ways hands-on, minds-on CT can be incorporated into learning experiences. The examples throughout the remainder of this paper also include rich active learning ideas for CT learning and integration.

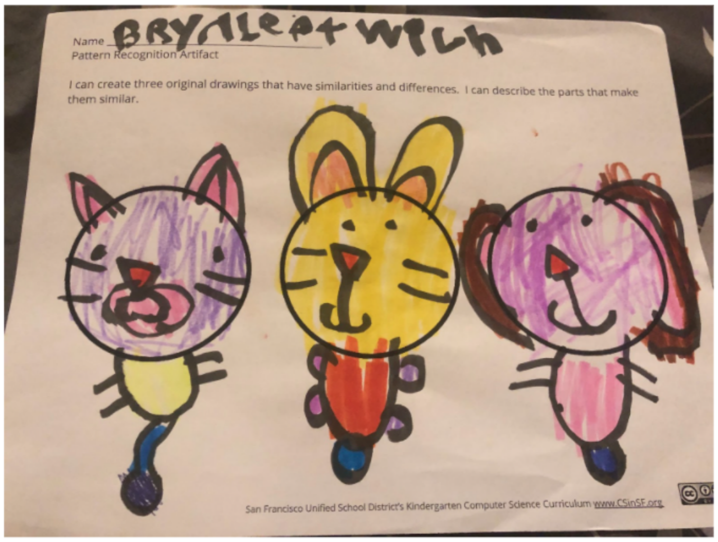
Tools for Teaching Elementary Computational Thinking/Computer Science

There are many great tools that can be used to develop computational thinking. Studies have shown that elementary teachers find computer programs such as Scratch and other CS-focused devices to be both valuable (Clark et al., 2013) and accessible to both teachers and students (Lee, 2011). Israel et al. (2015) specifically indicated that even skeptical or reluctant teachers found value in teaching CS when provided with certain pedagogical tools. We have organized these into five categories (unplugged, plugged, tools, robotics/devices, curriculum, and books). We will briefly discuss examples here, but a more detailed description and coverage of additional tools can be found in the Appendix, which was developed by Guo and Fagundes (2020).

Unplugged Activities

Educators and parents often turn to unplugged CT activities when access to digital resources is difficult or there is a desire to focus on the content rather than a gadget or object. Unplugged CT activities lack a digital component and typically come in the form of games and activities or as curricular teaching materials. Curzon et al. (2018) described unplugged computing as “physical objects and role play are used to illustrate computing concepts” (p. 514). It seems that many of the early childhood educational experiences with computational thinking incorporated unplugged activities. The purpose of these seemed to be the intent to make abstract concepts more concrete (see Figure 5).

Example Kindergarten Patterns


Name
Pattern Recognition Artifact
I can create three original drawings that have similarities and differences. I can describe the parts that make them similar.
San Francisco Unified School District's Kindergarten Computer Science Curriculum www.CSinSF.org

Create a list of all the different features that the animals with the details abstracted out of the sentence.

- The animal has [insert type] ears.
- The animal has [insert type] legs.
- The animal has [insert type] tail.

Figure 5. Example of a Kindergarten Unplugged Activity from CS for All in SF.

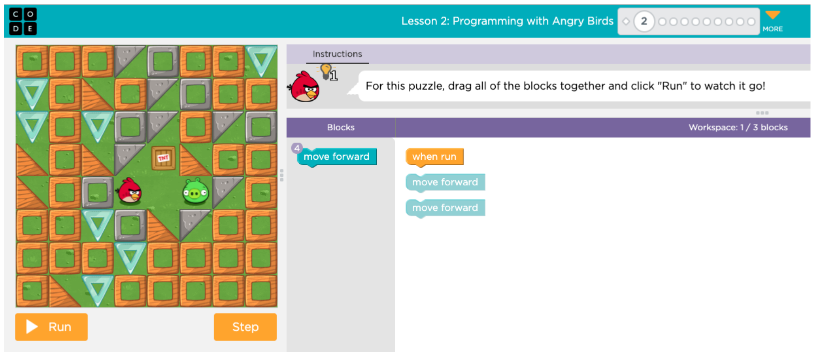
Researchers have examined the effectiveness of unplugged activities at the PreK-5 grade levels. In one example, Faber et al. (2017) implemented six 90-minute unplugged lessons around programming concepts (algorithms, variables, repetition, conditionals, and binary) in 26 primary schools in the Netherlands. Based on observations and interviews, they asserted that the unplugged games made the lesson more engaging for students. In a quasi-experimental study of 84 second-graders in Spain, del Olmo-Muñoz et al. (2020) explored the differences between unplugged and plugged CT activities. The control group of 42 students completed 3 unplugged CT activities and 3 plugged activities, whereas the experimental group completed 6 plugged activities. The results showed that those students who completed unplugged and plugged activities significantly outperformed students that had only completed plugged activities.

In general, it seems that unplugged activities are particularly successful for early childhood and primary students. Some of the most popular elementary CS curriculums at this point (Code.org's CS Fundamentals, Project Lead the Way's CS Launch, and Computer Science for All in SF) use both unplugged and plugged activities for CT/CS concepts. In a quasi-experimental study of 35 elementary students, Hermans and Aivaloglou (2017) provided one half of the students with four plugged lessons and the other half four unplugged lessons. Afterwards, both groups participated in four weeks of Scratch lessons. After eight weeks, there was no difference between the two groups with regards to their knowledge of programming concepts. However, the unplugged group was more confident of their ability and used a wider selection of Scratch blocks. This may show the importance of utilizing both unplugged and plugged activities to encourage students' confidence and creativity.

Typically, CS curriculum uses unplugged activities first, and then moves on to utilizing computational toys or visual block-based programming. Wohl et al. (2015) conducted a quasi-experimental study over three groups of 28 students aged five to seven. They explored the differences between the order of three different approaches to teaching CS concepts: unplugged computing, tangible computing (with Cubelets) and visual block-based programming (Scratch). Based on their observations, Wohl et al. (2015) found that after the unplugged sessions, students were the most engaged in CS concepts. They suggested that the "unplugged session seemed to demonstrate that young children can be introduced to and engaged in relatively complex ideas" (p. 5). This seems to be a developmentally appropriate way to build students' understanding, and then apply it with an external device.

Plugged Activities

Plugged activities consist of online puzzles or games that students can explore. They are often very tailored and heavily guided. Some of these include Code.org, Hour of Code activities, Kodables, and Tynker. These plugged activities guide students through a set of graded exercises/puzzles to move characters through a scene (for example, a red Angry Bird going after a Green Pig). Through these experiences, students learn some of the basics concepts around programming such as sequencing, loops, and conditionals by using command blocks like "move forward" or "turn left." These activities are less open-ended than other visual block-based programming environments like Scratch or Blockly.



For this puzzle, drag all of the blocks together and click "Run" to watch it go!

Figure 6. Retrieved from Code.org: https://studio.code.org/s/express-2020/stage/2/puzzle/2

Studies have shown that these plugged activities seem to increase young students' interest and conceptual understanding of coding (Pila et al., 2019). However, studies have also shown that young students were still unable to verbally explain coding (Pila et al., 2019). In a study various implementations of CT in math with preschool children, Lavigne et al. (2020) found that activity sessions with digital apps took longer than hands-on activities (17 mins as compared to 11 mins). In addition, teachers and students both spent the majority of their time focused on the learning goal 65% when working on unplugged activities and 71% when working on plugged activities (Lavigne et al., 2020).

Curriculum Example of Unplugged and Plugged Activities: Code.org

Perhaps one of the most common ways elementary teachers are teaching CS is through Code.org's CS Fundamentals curriculum. This curriculum incorporates both unplugged and plugged activities, typically introducing a concept with an unplugged activity and following with a plugged activity where the student applies and practices the new concept. The concepts are heavily focused on computer science, but the detailed lesson plans also provide links to potential literacy, math, and other relevant K-5 national content area standards. There have been several studies conducted to investigate the impact of the Code.org curriculum. Kalelioğlu (2015) investigated the impact of Code.org on 32 4th year Turkish students' reflective thinking skills through a quasi-experimental design. Although there were no significant differences in students' reflective thinking skills, students developed a positive attitude towards programming and female students were as successful as their male counterparts. In another study, Lambić et al. (2020) examined the impact of Code.org's second course on 293 seven to ten year-olds. They found that older students reported a significantly more positive attitude towards programming than younger students. The authors observed that many of the younger students were unable to solve many of the programming tasks, which resulted in a negative attitude towards programming. Therefore, selecting appropriately challenging materials is critical to students developing positive attitudes towards CS.

One of the authors of this paper is a Code.org CS Fundamentals trainer for the state of Indiana. Although there are some exceptional pieces of Code.org's work (including wonderfully detailed lessons plans, videos, widgets and materials), one of the challenges I have noticed is that K-5 teachers often only use Code.org's plugged lessons. Without implementing the unplugged lessons, students are lacking the stronger introduction to concepts and CS becomes equated to playing games and solving puzzles online. Elementary teachers often describe the challenges of fitting CS into the school day (Ozturk et al., 2018). Therefore, Century et al. (2020) worked to embed Code.org into one school district's non-negotiable elementary literacy block. They developed "Time for CS" (Time4CS) modules that included science, ELA, and social studies lessons and associated Code.org lessons connected with a problem-based theme. During the 2016-2017 academic year, 157 teachers implemented two modules for each grade (3rd – 5th) during existing 180-minute literacy blocks. The teachers who used Time4CS modules implemented more CS lessons than other in their district. In addition, the study found that these higher amounts of interdisciplinary teaching practices were associated with higher student achievement, specifically students' state assessment ELA scores. Finally, the approach of incorporating Code.org Fundamentals within existing curriculum seemed to present a more feasible way to provide more CS opportunities to 3rd – 5th grade students. They stressed that this study proves that "it is possible to make time in the elementary school day for CS, and that there are no negative consequences for core subjects (e.g., ELA and mathematics)" (Century et al., 2020, p. 1). Based on our engagement with this curriculum, we recommend this as an excellent starting point for teachers, but stress the importance of reading the lesson plans and implementing unplugged lessons before having students begin the plugged sessions.

Robotics/Devices

There are a wide range of robotics/devices that have been utilized to incorporate computational thinking into elementary classrooms. Some robotics have been designed to feature push button coding, claiming that this process enables younger students who may not have the capacity to utilize coding apps to engage in simple commands structures and coding.

Bee-Bots

One of the more popular push-button coding devices for younger children is Bee-Bot (Figure 7). Bee-Bot is a friendly looking bumblebee that has four arrows on its back enabling it to move forward 6 inches, backwards 6 inches, and turning to the left or right. PreK-5 teachers have students use Bee-Bot with a large grid mat. Students must code the Bee-Bot to arrive on certain numbers, letters, or shapes. This activity enables teachers to integrate computational thinking into their other subject areas and make it engaging for students. A few studies have shown that early childhood students (preschool and kindergarten) show increase CT skills after using Bee-Bots (e.g., Caballero-González, Moñoz, & Muñoz-Repiso, 2019; Papadakis & Kalogiannakis, 2020).

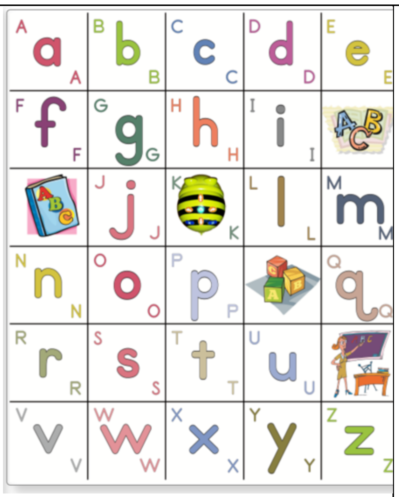
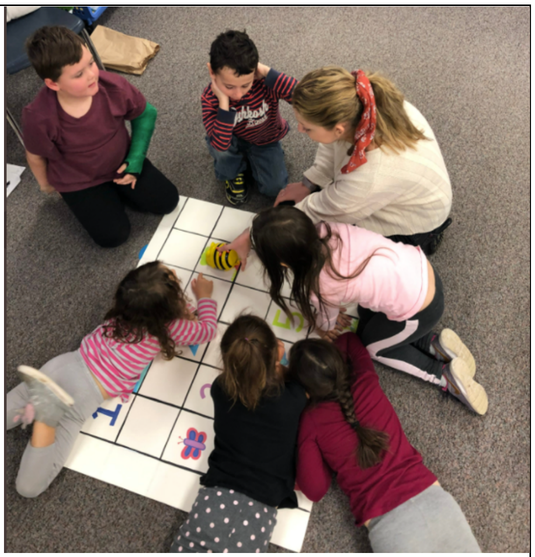
Picture from Bee-Bot website.	 Picture from Bee-Bot website.	 Picture from @anneleftwich

Figure 7. Example of Bee-Bot robots and an activity for kindergarten students.

Robot Mouse

Another popular robotic device that is used with early elementary students to develop CT skills is the Robot Mouse. The robot mouse is sold separately or within the Code and Go™ Robot Mouse Activity Set developed by Learning Resources. The entire activity is a game in which the player sets up steps for the robot mouse to follow through a physical maze to arrive at the cheese. The set include Colby, the programmable battery-operated robot mouse, a wedge of cheese that when Colby touches it, his nose lights up and he makes a cooing noise, 16 square tiles that can be interlocked to make the floor of the physical maze, gates and tunnels to add to the maze, code cards printed with directional arrows or action symbols, and maps of maze puzzles for the player to solve. Figure 8 shows children playing with the Code and Go Robot Mouse activity set. Moore, Brophy et al. (2020) used the robot mouse game as a way to study students ability to translate between coding representations.

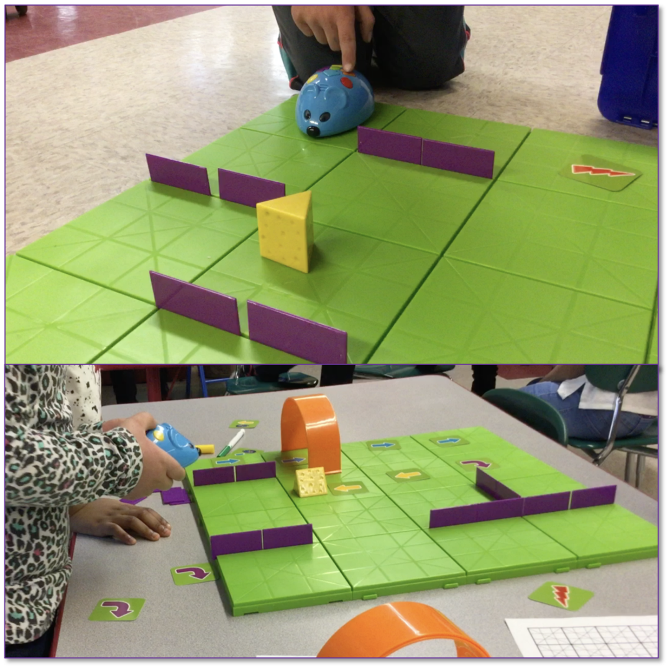


Figure 8. Students coding the robot mouse to navigate the maze in order to get to the cheese.

This particular robot has an add-on expansion set to specifically target early math concepts. The kit includes number and coding cards, dice, and a playmat to introduce coding lessons with addition, subtraction, and number sequencing. The activity guide includes lessons and games to integrate the coding mouse into a range of math lessons (e.g., even and odd numbers).

Robots and robotic devices are popular STEM toys for kids. These devices have a lot to offer as both free-choice play and devices to help scaffold learning in more formal environments. Furthermore, robotics can also include "the learning of computer programming concepts such as iteration, input/ process/output, and control structures (procedural flow)" (Sullivan & Heffernan, 2012, p. 107). There are many additional robotic devices that we have not had a chance to review for this paper. See the Appendix for a more comprehensive list of robotic devices.

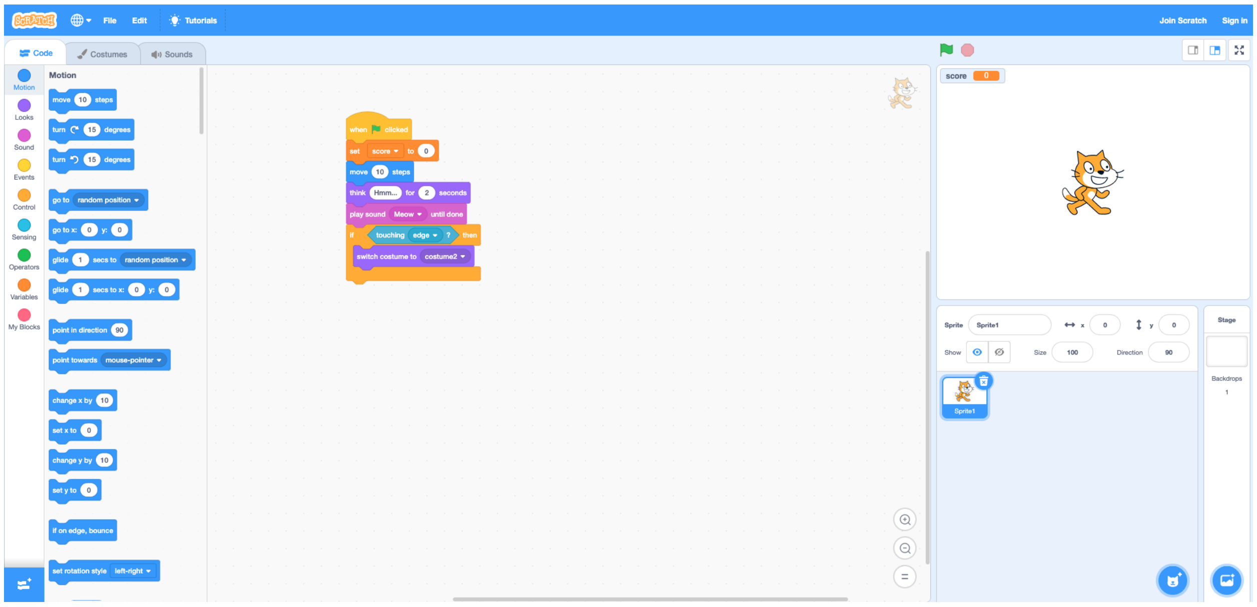
Robots and robotic devices have potential to capture the imagination of young learners. However, the context in which robotics is introduced can impact who will be engaged. Studies have shown that robotics may not appeal to female students at the secondary levels, and female students may need more support and be less confident (Sullivan & Bers, 2019). There is a lack of research on whether this also applies at the elementary levels. Therefore, special attention should be dedicated to ensure all students are being engaged by robotics inspired learning experiences.

Open-Ended Tools

The resources mentioned in this category employ open-ended apps that students can use and explore in a wide range of ways. Perhaps the most common is visual block-based coding applications/software. The most popular visual block-based coding applications are ScratchJr and Scratch. According to Bers (2018a), Scratch was designed to “provide easy ways for novices to get started (low floor), ways for them to work on increasingly sophisticated projects over time (high ceiling) and multiple pathways for engagement for all children with diverse interests (wide walls)” (p. 2). In a review of CT education studies, Lye and Koh (2014) identified nine studies that examined how programming was incorporated into the K-12 curriculum. Most (n=8) of these studies utilized Scratch or Logo.

Scratch

Scratch is an open-ended coding tool that utilizes visual blocks to enable students to program a character. It can incorporate media and can be interactive. Students have used Scratch to create a wide range of projects including animated stories, news shows, music/arts projects, simulations, tutorials, book reports, and much more. Scratch was designed to support students (ages 8-16). Scratch is available in many different languages. Blocks are organized into categories based on actions and color-coded to make things easier to find on the far left-hand side. The middle is where students build the code using blocks and the stage on the right-hand side where the characters enact the coded program (Maloney, 2010)(see Figure 9).



```
when clicked
set score to 0
move 10 steps
think Hmm... for 2 seconds
play sound Meow until done
if touching edge ? then
    switch costume to costume2
```

Figure 9. Example of Current Scratch Interface. Screenshot taken from https://scratch.mit.edu/projects/editor/?tutorial=getStarted

Zhang and Nouri (2019) reviewed all empirically based Scratch articles at the K-9 grade levels between 2007 and 2018. They applied the Brennan and Resnick (2012) CT framework to analyze and identify what concepts were being researched. They found that out of 55 articles, the CT concepts that were most commonly addressed were loops (n=28 studies), sequences (n=26), and conditionals (n=24). These have been identified as the basic control structures in all programming languages (Zhang & Nouri, 2019). There were at least ten studies that also investigated the CT concepts of variables, coordination, reading code, Boolean logic/operators, parallelism, events, and abstraction. In addition, nine studies focused on the CT practices abstracting and modularizing, while eight studies focused on debugging and testing. Zhang and Nouri (2019) found that K-9 students' uses of Scratch which typically fall into games (n=21), animation (n=5), or storytelling (n=4). When examining the subject concepts, Zhang and Nouri (2019) found that 64% of the 55 K-9 Scratch studies were situated within computer science, 12% in arts/music/crafting, 11% in language, 9% in math, and 4% in science.

Studies have shown that Scratch has enabled primary students to build CT skills (Zhang & Nouri, 2019), increase students' attitudes and interest in computer science (Sáez-López et al., 2016) and even digital competencies and 21st century skills (Nouri et al., 2020). Sáez-López et al. (2016) implemented a two-year long intervention where 107 5th and 6th grade students used

Scratch within art and social science contexts (see evidence here: https://scratch.mit.edu/studios/804018/). The lessons were based on the Creative Computing Curriculum for Scratch created by the Creative Computing Lab at the Harvard Graduate School of Education. Sáez-López et al. (2016) stated that this interdisciplinary approach seemed to motivate students. They recommend that this approach to develop students' computational thinking within a curricular context can impact students' CT skills, art and history skills, and motivation for participating in more active styles of learning.

Wohl et al. (2015) conducted a quasi-experimental study over three groups of 28 students aged five to seven. They explored the differences between the order of three different approaches to teaching CS concepts: unplugged computing, tangible computing (with Cubelets) and visual block-based programming (Scratch). Based on their observations, after the Scratch sessions, the students encouraged the students' creativity, although it was difficult for the students to use. This was likely due to the fact that Scratch was too difficult for younger students to work with.

ScratchJr

ScratchJr programming software was created by the authors of Scratch for younger students in kindergarten to second grade. Design considerations included developmentally appropriate interface and methods of interaction. Specifically, the authors mentioned creating software that had a "low floor and (appropriately) high ceiling, wide walls, tinkerability, …and conviviality" (Flannery et al., 2013). Flannery et al. (2013) examined how younger students engaged with Scratch in a small pilot study of kindergarteners through second graders. Kindergarteners through second graders struggled to use Scratch due to literacy capabilities (unable to read Scratch block), lack of understanding measurements (grid patterning was difficult), and lack of ability to think abstractly and predict the results of blocks. Therefore, they set out to develop a version that could meet younger students' needs.

After designing Scratch Jr, Flannery et al. (2013) tested the software with 100 kindergarten through second grade students in nine sessions. Although all students were able to use the software to create programs, kindergarteners seemed to struggle with the interface. One kindergarten project used Scratch Jr to talk about basic motions (over, under, etc.) while the first and second grade students often used it for retelling stories, utilizing multiple characters and actions. Flannery et al. (2013) also discussed the importance of building curriculum guides that could be integrated into math and literacy at the K-2 levels. They developed an online community where early childhood educators could post and share materials and lessons (Flannery et al., 2013).

Bers (2018a) described that the programming blocks were organized into six categories, signified by different colors, to describe overall coding constructs: “yellow Trigger blocks, blue Motion blocks, purple Looks blocks, green Sound blocks, orange Control flow blocks, and red End blocks” (p. 2). Students are able to connect blocks together to control the characters. They can create their own characters and background. The program was designed to be used like a narrative structure with different pages established to mimic the creation of a book by establishing pages and containing a beginning, middle, and end. In addition, students can integrate text and speech bubbles into their projects.

The authors of ScratchJr wanted to create a digital playground with this app. They have pointed out that the application lacks the collaborative elements that are typically found on a playground between children. Therefore, they organized a DevTech Research Group that has created a Collaborative ScratchJr Projects Guide. This guide supports teachers in collaborative projects for students that can incorporate moving characters across multiple screens and iPads (Bers, 2018a)(see Figure 10 for example of collaborative Scratch Jr projects).

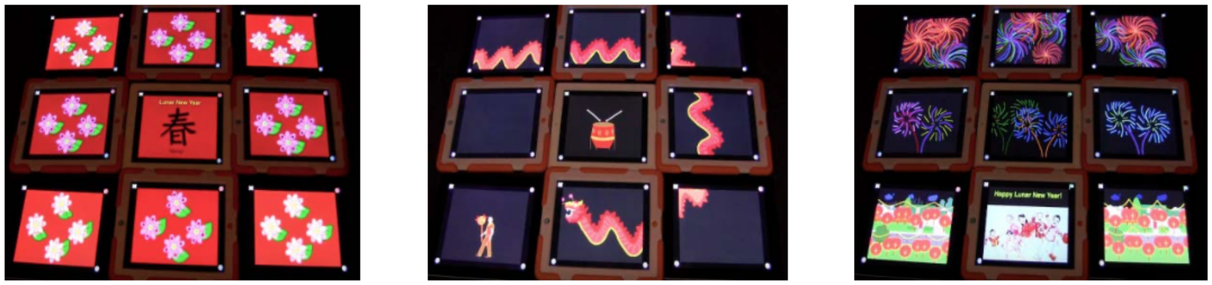


Figure 10. Example of Scratch Jr Collaboration Project from Bers (2018a).

LEGO WeDo 2.0

Other tools/resources in this category include other robotics that utilize block-based coding. These include, but are not limited to Qubo, LEGO WeDo 2.0, WonderWorkshop’s Dash ‘n Dot, LEGO Boost, Artie 3000, and VEX Robotics. LEGO WeDo 2.0 has a connected curriculum that integrates CS into engineering and science concepts. For example, one activity has students investigating pollination. Students build and program a pollination model using

LEGOs and a block-based coding app (see Figure 11 for example from LEGO WeDo 2.0's website).

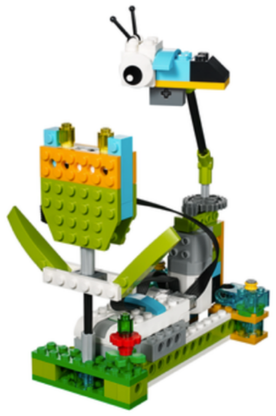
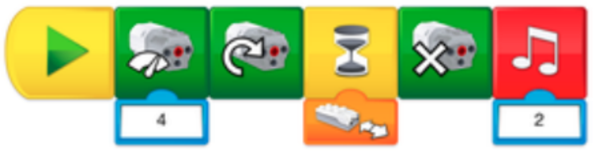
	

Figure 11. LEGO WeDo 2.0 Examples. (https://education.lego.com/en-us/lessons/wedo-2-science/plants-and-pollinators#3-create-phase)

Chalmers (2018) examined how UK teachers in grades one through six implemented LEGO WeDo 2.0 activities. Teachers implemented lessons around pulling (investigating balanced and unbalanced forces), speed (investigating factors that make a car accelerate), structures (investigating characteristics to make buildings earthquake resistant), and plants and pollinators (modeling the relationship between pollinators and flowers). Based on these implementations, teachers reported that students seemed to develop computational thinking concepts (sequencing, loop, and pattern recognition) and practices (problem solving and debugging). Teachers also reported that students seemed to develop CT perspectives, such as persistence and iterating on designs, as well as collaborating and sharing ideas with their classmates (Chalmers, 2018).

LEGO WeDo 2.0 seems to be a good introduction to CT/CS ideas that are integrated into a problem-based situation. Like many existing curriculums, it is important to show how all these concepts build on each other. Otherwise, it can often seem like small STEM activities, rather than a curriculum. We could envision this as an introduction project to a science idea/topics like investigating forces, or as a culminating activity to apply concepts they have already learned.

Cozmo

One of the more sophisticated robots is Cozmo. Cozmo is a robot that develops as you play with it. It has been used as an introduction to artificial intelligence for elementary-aged students. The Cozmo robot uses Calypso (a scaffolded robot programming environment). The programming environment allows students to program Cozmo with advanced features such as "visual recognition of objects and faces, simultaneous localization and mapping (SLAM), landmark-based navigation, and speech input" (Touretzky & Gardner-McCune, 2018, p. 1). In one study, Ehsan, Cardella, and Hynes (2020) examined two children with autism (8-10 years old) that tried Cozmo's Ambulance activity with their mothers. Although both were able to code the robot and showcased multiple CT competencies, their experiences were different based on the interactions with their mothers. One mother/child interaction resulted in the child's CT problem-solving and completion of the given challenges. The other mother/child interaction was less successful as the mother tended to focus on the child's deficits. Ehsan et al. (2020) concluded that "All children can engage in CT competencies if the adults working with them focus on their strengths and potential rather than their deficits, and accordingly appropriate guiding strategies and learning opportunities are provided" (p. 7).

Curricula

There are a wide range of CS-focused curricula available for PreK-5. Throughout the paper, we have brought in many examples of curricular innovations that highlight CT in different ways. To supplement these examples, we also bring forth one additional fairly comprehensive resource. According to CSforAll's content provider membership, there are currently 152 different curriculum providers for PreK-5th grade (http://bit.ly/CSforAllPreK5). These range from CS specific (i.e., Codelicious), to robotics (e.g., Exploring Robotics), to AI-focused (e.g., AInspire), to STEAM integrated CS (e.g., SAM Labs). There are also resources for teachers such as "No Fear Coding K-5". In this resource, the author walks teachers through integrating Bee-Bots, Code.org lesson, and Scratch across the curriculum. One of the most commonly used CS-focused curriculum is Code.org's CS Fundamentals curriculum (described earlier). While we did not survey the majority of curricular resources, we are happy to see that more and more resources that integrate CT in meaningful ways are available to teachers and students.

Books

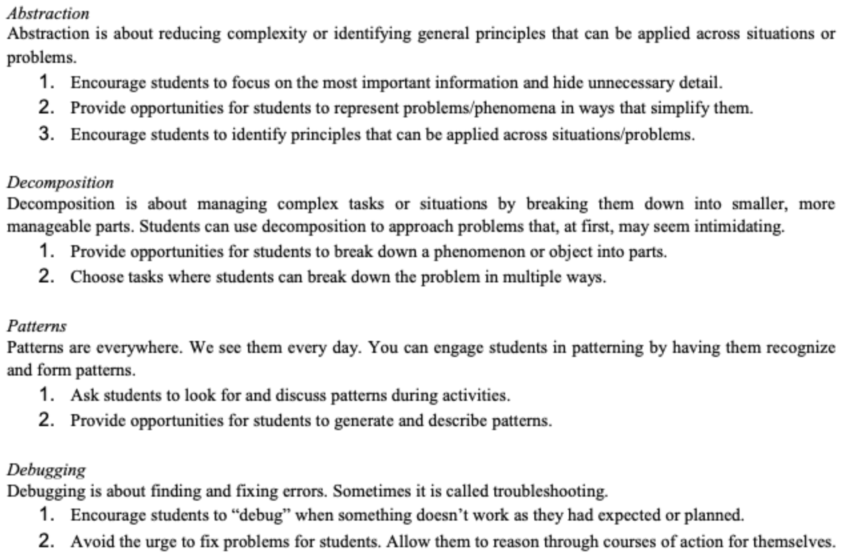
There have also been a range of different children's books focusing on computer science within the past few years. Specifically, these books tend to focus on using literacy to teach CS/CT ideas and principles. Haroldson and Bellard (2020) reviewed 45 picture books and graphic novels published between 2015 and 2019 that focused on CS at the K-8 grade levels (see below for the list from p. 7-8). The authors investigated what the computer science practices that the characters in the books engage in. Using the four main CS practices established in the K-12 CS Framework, the authors reviewed the books for evidence of one of the four CS practices. They found that 70% of the books contained at least three of the seven practices, with only two-percent of the books covering all seven practices. The most common practices addressed were Creating Computational Artifacts (80%), Developing and Using Abstractions (67%), and Recognizing and Defining Computational Problems (58%). The least frequently addressed practice was Fostering an Inclusive Computing Culture (9%). Books to support CT/CS integration will be discussed more in the literacy integration section.

Integration of CT/CS into Other Elementary Subjects

Scholars have argued that computational thinking is intrinsic to all subject areas, describing it as the core of all modern disciplines (Henderson et al., 2007). Lavigne et al. (2020) suggested that "developing computational thinking (CT) skills at a young age is critical for preparing preschool children to engage with the technologies that have become central to nearly every occupation and for improving achievements in STEM, literacy, and other disciplines" (p. 63). Some scholars have pointed out that computer science shares similar content and inquiry methods with science (Fluck et al., 2016) or math (Rich et al., 2019). In science, Fluck et al. (2016) described that the concept of data and analysis in CS overlaps with the concepts of observing phenomena and proposing hypotheses in the scientific method. In math, Rich et al. (2018) explained that terms embedded in computational thinking (e.g., algorithm) and programming (e.g., variables) share similar terms embedded in mathematics.

Some have claimed that by integrating computer science concepts with the other disciplines, it could promote the innovation of computer science curricula (Sahami et al., 2013) or could help with problem-solving by opening multiple ways of thinking (Denning et al., 2017). However, scholars also state that embedding the concepts of Computer Science within the other disciplines through the concept of computational thinking greatly challenged teachers (Barr & Stephenson, 2011). Grover and Pea (2013) pointed out that more research was needed to figure out how to integrate CT into elementary subject areas.

As the integration of CT into other subjects areas has proved challenging for teachers to accomplish, Yadav et al. (2019) provided a toolkit for teachers to incorporate CT into their classrooms using four concepts: abstraction, decomposition, patterns, and debugging (see Figure 12 for a breakdown of the entire guide from their article). For abstraction, teachers should focus on reducing complexity, encouraging students to simplify and focus on the more important information. For decomposition, teachers need to help students break down problems into smaller, more manageable parts. For patterns, facilitate opportunities for students to recognize and create patterns. For debugging, teachers can focus on encouraging students to identify the errors in their work, and to fix it themselves.



Abstraction

Abstraction is about reducing complexity or identifying general principles that can be applied across situations or problems.

1. Encourage students to focus on the most important information and hide unnecessary detail.
2. Provide opportunities for students to represent problems/phenomena in ways that simplify them.
3. Encourage students to identify principles that can be applied across situations/problems.

Decomposition

Decomposition is about managing complex tasks or situations by breaking them down into smaller, more manageable parts. Students can use decomposition to approach problems that, at first, may seem intimidating.

1. Provide opportunities for students to break down a phenomenon or object into parts.
2. Choose tasks where students can break down the problem in multiple ways.

Patterns

Patterns are everywhere. We see them every day. You can engage students in patterning by having them recognize and form patterns.

1. Ask students to look for and discuss patterns during activities.
2. Provide opportunities for students to generate and describe patterns.

Debugging

Debugging is about finding and fixing errors. Sometimes it is called troubleshooting.

1. Encourage students to "debug" when something doesn't work as they had expected or planned.
2. Avoid the urge to fix problems for students. Allow them to reason through courses of action for themselves.

Figure 12. Toolkit screenshot from Yadav et al. 2019

The Importance of Context

When considering CT integration into other subjects, the contexts that students work within are highly important. A theme that runs through much of the STEM integration literature is that STEM activities should be focused on realistic or real-world problems (Moore, Johnston, & Glancy, 2020). The contexts used in CT integration activities should represent complexity of real-world problems (Angeli et al., 2016; Berland & Steingut, 2016). Connecting the CT lessons to other school subjects - particularly STEM subjects (Lesh & Harel, 2003; Ryan et al., 2017), STEM careers (Ryu, Mentzer, & Knobloch, 2018), and the community - making them more socially and culturally relevant (Johnson, 2013) - are all potentially good ways to help students make connections in their learning of CT. Furthermore, it has been argued that embedding STEM+C content in real-world contexts makes students more motivated and engaged in the learning because they are more meaningful and relevant to students' lives (Angeli et al., 2016; Berland & Steingut, 2016; Guzey et al., 2016).

Integration of CT/CS into Elementary Engineering and Engineering-Based STEM

Engineering and CT are inherently connected (National Research Council [NRC], 2011) . Computer science is often considered one of the disciplines of engineering and in many universities, the CS department is within the college of engineering. In fact, computer science, software engineering, and computer engineering overlap significantly - with nuanced differences in the focus of each discipline. The hallmark of engineering design is that the engineer designs a technology to meet a need. Since computer programs are technologies, then necessarily one who designs a computer program would also use the principles of engineering design to design it. Brennan and Resnick (2012) identify those who code as designers. Furthermore, criteria and constraints guide and limit how designers work toward their end product (NRC, 2010). Ehsan, Cardella, and Svarovsky (2018) found that CT elements were present within in the engineering design process employed by young students. Table 1 provides an overview of how their research identified overlaps between CT competencies and engineering design. While this is not a comprehensive understanding of how engineering and CT overlap, it does demonstrate the potential for the meaningfulness and thoroughness of how these two areas can work together.

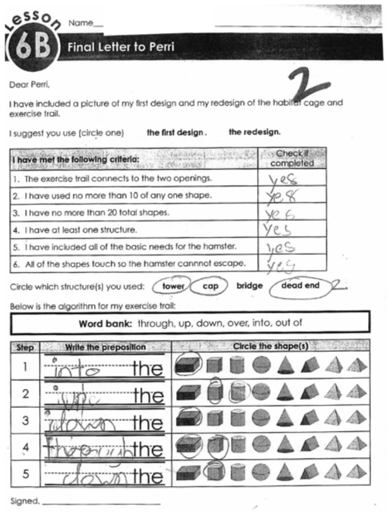
Table 1

Engineering design elements and where CT competencies overlap (Ehsan, Cardella, & Svarovsky, 2018)

Engineering Design Element	Computational Thinking Concept Used During Design Task
Problem Scoping	Data Collection Abstraction Problem Decomposition
Generating Ideas	Data Analysis Abstraction Problem Decomposition Pattern Recognition

Idea Selection	Pattern Recognition Abstraction
Testing Solution	Simulation Parallelization Debugging/Troubleshooting Pattern Recognition

Engineering design-based integrated curricular programs are a great way to introduce students to CT constructs. Two such programs are PictureSTEM and ETA Hand2Mind: STEM in Action. PictureSTEM (picturestem.org) is a set of three curricular units for grades K-2 that integrate STEM+C content using engineering contexts and picture books. Each unit focuses on at least one mathematics, science, and CT idea that is standards-based for that grade in service of solving an engineering problem for a client (Tank et al., 2018). These curricula are a result of three NSF-funded projects (#1442416, #1519387, #1543175). Unlike many of the other curricula we have found, the CT is integrated into a larger unit as a means to solve an engineering problem that the client for the problem needs solved (Hynes et al., 2019). For example, in *Designing Hamster Habitats* (first grade), the students are working for Perri who is the owner of Perri's Pet Palace. Perri wants a new design for the exercise trail for her hamster habitats. To integrate CT into this, the students must present their design with an accompanying algorithm that instructs Perri and her customers how the hamster will move through the habitat. See Figure 13 for an example. In order for students to be able to learn about algorithms prior to this final artifact they must present to Perri, students are introduced to algorithms through a picture book, *Joey and Jet* by James Yang, and then through following and creating algorithms with tangrams, which is an extension of some the mathematics integration already occurring in the unit. In the Joey and Jet reading lesson, students are asked to retell the story through sequencing using flowcharts, which is both a literacy standard and a CT concept (Figure 14). Then the paired CT lesson has students using tangrams to first follow algorithms then create their own (Figure 15). Each unit in the PictureSTEM curricula has a similar manner in which CT is integrated into the whole engineering design project and paired lessons that highlight the literacy and CT integration. The kindergarten PictureSTEM unit focuses on pattern recognition and abstraction in literacy and engineering through basket weaving and the second grade unit focuses on sequencing, debugging, and algorithm development through use of a robotic device (Hynes et al., 2019; Tank et al., 2018). These developmentally appropriate introductions to CT work well as jumping off points for further explorations into CT for K-2 students.



Lesson 6B Name

Final Letter to Perri

2

Dear Perri,

I have included a picture of my first design and my redesign of the habitat cage and exercise trail.

I suggest you use (circle one) **the first design.** **the redesign.**

I have met the following criteria:	Check if completed
1. The exercise trail connects to the two openings.	yes
2. I have used no more than 10 of any one shape.	yes
3. I have no more than 20 total shapes.	yes
4. I have at least one structure.	yes
5. I have included all of the basic needs for the hamster.	yes
6. All of the shapes touch so the hamster cannot escape.	yes

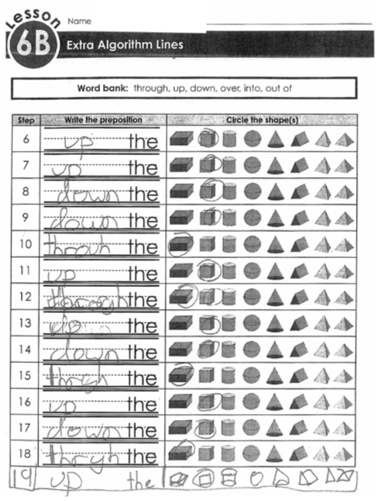
Circle which structure(s) you used: [✓ tower] [✓ cap] bridge [✓ dead end] 2

Below is the algorithm for my exercise trail:

Word bank: through, up, down, over, into, out of

Step	Write the preposition	Circle the shape(s)
1	into the	
2	up the	
3	down the	
4	through the	
5	down the	

Signed, ____________



Lesson 6B Name

Extra Algorithm Lines

Word bank: through, up, down, over, into, out of

Step	Write the preposition	Circle the shape(s)
6	up the	
7	up the	
8	down the	
9	down the	
10	throuh the	
11	up the	
12	through the	
13	up the	
14	down the	
15	throgh the	
16	up the	
17	down the	
18	thrygh the	
19	up the	

Figure 13. Students developing the algorithm for the hamster habitat trail and the resulting final letter to the client that includes the algorithm.

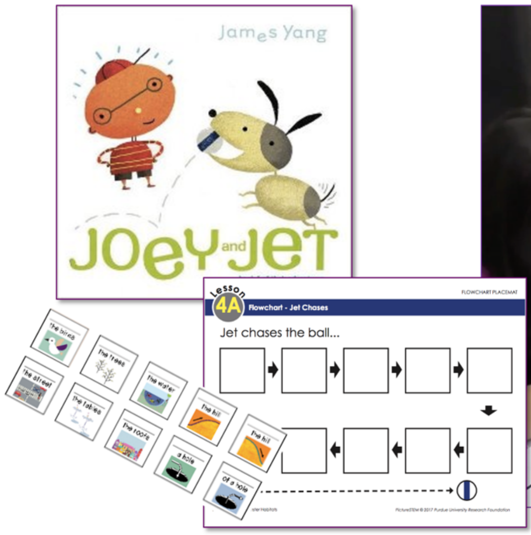
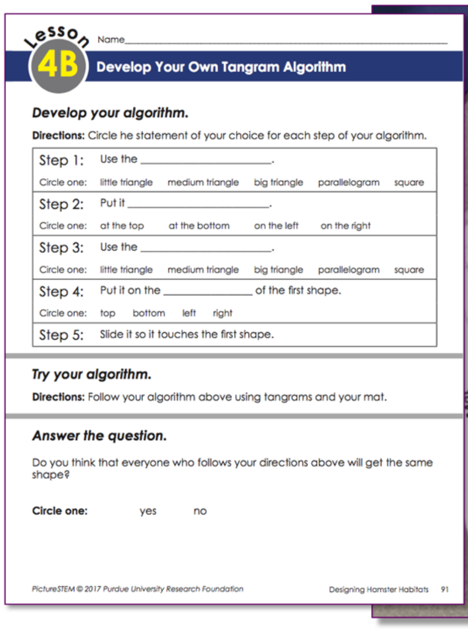


Figure 14. *Joey and Jet* flowchart for sequencing. Student retell the story through use of prepositions and putting the game of fetch in the order that the dog, Jet, chases the ball.



Lesson 4B Name____________________

Develop Your Own Tangram Algorithm

Develop your algorithm.

Directions: Circle he statement of your choice for each step of your algorithm.

Step 1:	Use the ________________.
Circle one:	little triangle medium triangle big triangle parallelogram square
Step 2:	Put it ________________.
Circle one:	at the top at the bottom on the left on the right
Step 3:	Use the ________________.
Circle one:	little triangle medium triangle big triangle parallelogram square
Step 4:	Put it on the __________ of the first shape.
Circle one:	top bottom left right
Step 5:	Slide it so it touches the first shape.

Try your algorithm.

Directions: Follow your algorithm above using tangrams and your mat.

Answer the question.

Do you think that everyone who follows your directions above will get the same shape?

Circle one: yes no

PictureSTEM © 2017 Purdue University Research Foundation Designing Hamster Habitats 91

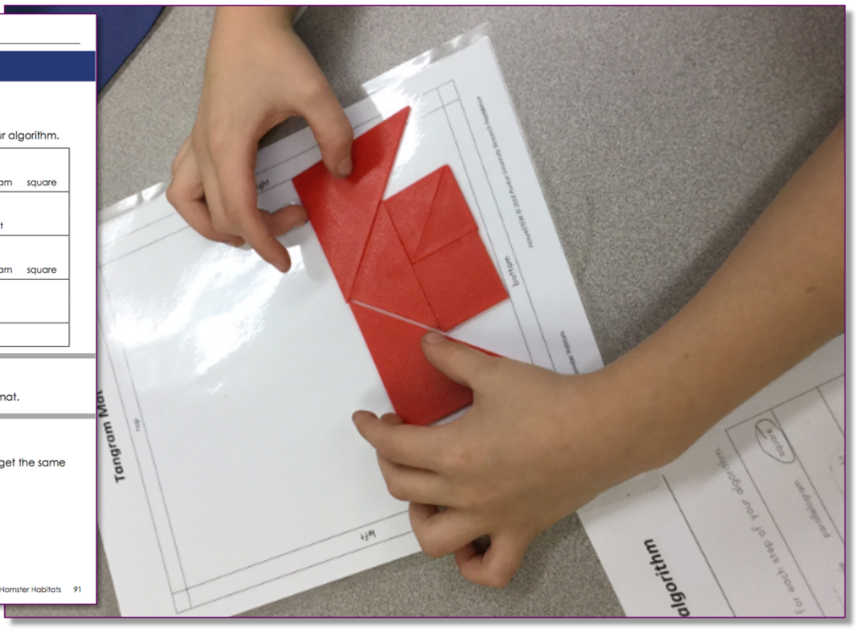


Figure 15. Student following an algorithm using tangrams (right). Students then create their own algorithm using the worksheet (left).

ETA Hand2Mind: STEM in Action curricula have two CT focused STEM kits. Each of these have a computational device (Robot Mouse and Botley) as part of a larger engineering design - based STEM integration project. In the *Coding Mouse Exploration*, students work to develop a program for the robot mouse that demonstrates that they understand the basic needs of mice (Figure 16). In the *Coding and Mineral Collection Challenge,* students develop a program for a robot that will collect unsafe minerals that have come to the surface of the earth (Figure 17). For both STEM in Action units, the CT concepts include code tracing and writing and debugging code.

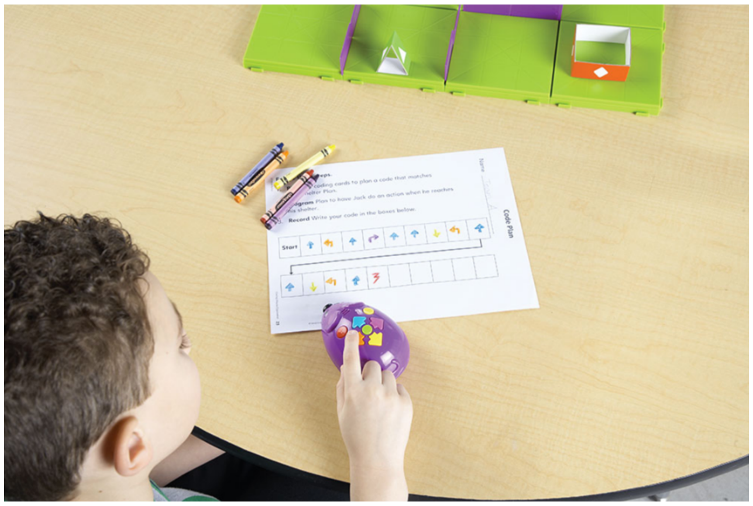


Figure 16. Child programming the robot mouse after designing code for the course. Image taken without permission from https://www.hand2mind.com/item/stem-in-action-coding-mouse-exploration/9123

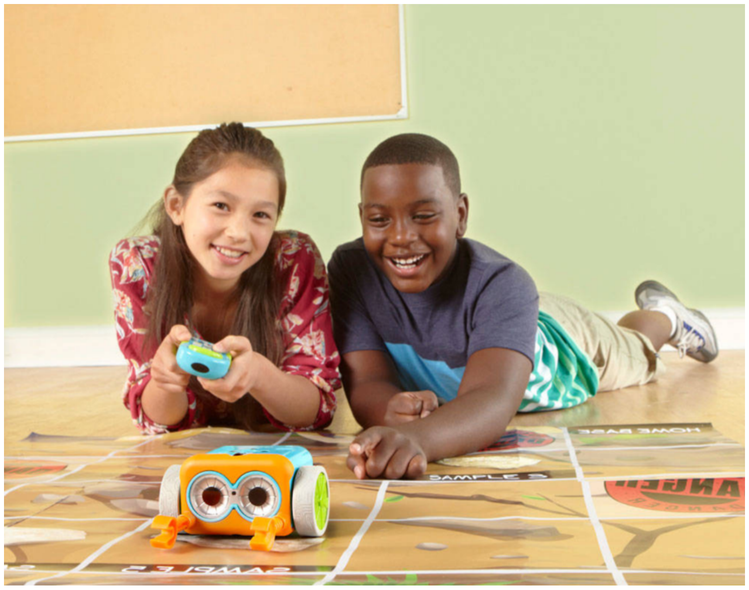


Figure 17. Children using Botley to collect mineral samples as they explore the surface of the earth. Image taken without permission from https://www.hand2mind.com/item/stem-in-actionreg-coding-mineral-collection-challenge/14422

Engineering design-based STEM integration has the potential to help students understand how CT and computing ideas are not only for the development of technological devices but also for using such devices for other important reasons that are helpful to people. Context is an important motivator for students as they engage with CT (Breiner et al., 2012; Corlu & Aydin, 2016; Guzey et al., 2016; Hsiao et al., 2019; Johnson et al., 2016; Milesi et al., 2017; Stubbs & Myers, 2015). Putting students in real world applications of CT will help them build an appreciation for CT beyond the strict constructs as we define them.

Integration of CT/CS into Elementary Mathematics

Some have suggested that mathematics and computational thinking are a natural fit for elementary integration (e.g., Rich et al., 2020). In fact, several studies have shown that mathematics achievement at the elementary levels seems to be linked to students' CT skills. Rich et al. (2020) provided a rich comparison of the interplay of CT and mathematics play out in standards. They found that precision, completeness, order, repetition, and conditionals are all part of both mathematics and CT - but have similarities and differences that need attention.

Studies of students at the middle school level have found that success in learning to think computationally can depend on mathematics ability and prior CT experiences both in and out of school (Grover, 2016). This finding was consistent with a few studies at the upper elementary level. Lewis (2012) found that 5th grade student performance on Scratch programming quizzes in a summer camp were highly correlated with their scores on a standardized math test. Salac et al. (2020) found similar results with 296 4th grade students who received instruction on events, sequence, and repetition based on the Creative Computing Curriculum. Although all students showed increases in their CT knowledge, there were statistically significant differences in learning outcomes between students with below grade-level math proficiency and those who were at or above grade-level. In a preschool study, Lavigne et al. (2020) studied an integration of CT ideas into mathematics instruction. The activities focused on cross-disciplinary concepts such as patterns, combining shapes into larger shapes, and sequencing. Lavigne et al. (2020) stated that "the fact that classroom teachers spent the majority of activity time on the target CT learning goals suggests that the approach to integrating CT into preschool math instruction shows promise" (p. 73).

Scratch and Scratch Jr. have been a focus of several interventions intended to integrate mathematics and CT. Scratch's Cross-Curricular Integration Guide includes many examples and resources on how to integrate Scratch and mathematics. Some of the 3rd - 8th grade project examples include building a multiplication game, creating a simple calculator, estimation game, fibonacci sequence, probability dice roll games, making a shape calculator, and fractions microworld. In another approach to integrating Scratch into mathematics, Maya Israel and Diana Franklin created *Action Fractions,* which provides 10-12 hours of math and Scratch instruction focused around 3rd and 4th grade fractions instruction (https://www.canonlab.org/actionfractionslessons).

Flannery et al. (2013) also described that Scratch Jr. was designed to be integrated with mathematics at grades K-2. In mathematics, ScratchJr can be used to support number sense and measurement (distance, rotation, time, and iterations). There is a removable grid of 20 by 15 squares, and students can use the grid to measure how far a character can/should move. Scratch also supports primary math at the upper elementary levels with its coordinate system, grid, x-axis, and y-axis.

Integration of CT/CS into Elementary Science

There is little research on integrating CT/CS into science at the elementary levels, and much of what is there studies the teachers rather than student learning. One of the few articles that focuses on student learning is from Dickes et al.(2020). This study looked at student learning within a model-eliciting activity (Lesh & Doerr, 2003) as a means to integrate science (kinematics and ecology) with mathematics (geometry) and CT for 3rd- and 4th-grade students. Within the kinematics portion of the integration, the students used an agent-based programming tool that is on the NetLogo platform to model footprints (relating measurement and motion). They found that the use of CT to model the phenomena required modeling cycles, or iterations. This led to iterative improvement of the students' representations (Dickes et al., 2020).

Even though the research base for integration of science and CT is not deep, curricular projects are still available to support CT and science integration. Scratch's Cross-Curricular Integration Guide included many examples and resources on how to integrate Scratch and science. Some of the 3rd - 8th grade project examples include using loops to create gravity systems, poison in America food nutrition, and animated biomes. There are also lesson plans and ideas associated with many of the CS/CT tools. For example, WonderWorkshop created a Dash Robot Life Cycles STEAM Project where students research their plan or animal, create a poster, and program Dash to visit each stage on the poster and talk about that stage. It should also be pointed out that many of the CS standards at the elementary level exist within the Science standards for different states (e.g., Indiana).

Integration of CT/CS into Elementary Literacy

Some have referred to computer science/computational thinking/computer literacy as our "new literacy" (Jacob & Warschauer, 2018; Shein, 2014; Wilson, 2013). In addition, Kelleher, Pausch, and Kiesler (2007) have suggested that computer science can be a modern storytelling mechanism at the middle school level. Also at the middle school levels, Burke and Kafai (2012) worked with ten students to draft, revise, and publish digital stories. Although students were able to learn CS programming concepts, they did not investigate the literary elements. Burke and Kafai (2012) caution that "digital stories in Scratch are likewise "products" that embody both the technical and the creative elements of composition and offer a broader conception of what "writing" with computers may look like in the 21st century" (p. 6).

Research has suggested that English ability at the elementary levels may be linked to students' CT skills. For example, studies at the middle school level have found that English ability is one of the contributing factors to students' success in CS learning (Grover et al., 2016). This finding was consistent with a study at the upper elementary level. Salac et al. (2020) found similar results with 296 4th grade students who received instruction on events, sequence, and repetition based on the Creative Computing Curriculum. Although all students showed increases in their CT knowledge, there were statistically significant differences in learning outcomes between students with below grade-level reading proficiency and those who were at or above grade-level.

There have been several ways literacy and CS have been used together in the curriculum. For example, in the younger grades, Scratch Jr was designed "to support narrative structure…[by letting] children create multi-page projects, like a book with a beginning, middle, and end. Text showing the name of each block can be revealed to support word recognition by letting children match intuitive icons with related text" (Flannery et al., 2013, p. 8). Lowe and Brophy (2019) examined the literacy practices of 18 K-2 students as they retold a fairy tale using Scratch Jr. Students seemed to struggle with creating animations that matched their drafted storyboards. There is little research documenting the role that texts, language, and vocabulary play into the development of a broader and more well rounded CS and computational thinking (CT) experience for students in K-5 classrooms.

There are several curriculums that have incorporated storytelling specifically in their CS curriculum. For example, CS in SF created an entire unit around storytelling with Scratch. In another example, Google for Education also developed CS First Curriculum, which has a specific unit dedicated to using Scratch to tell stories. Scratch's Cross-Curricular Integration Guide also included many examples and resources on how to integrate Scratch and English language arts. Some of the 3rd - 8th grade project examples include book reports, creation myths retelling, informative writing (water cycle), literature circles, parts of speech random sentence generator, and persuasive writing. In addition, Vicky Sedgwick has created curricular integration ideas using the Micro:Bit with English language arts. In another literary example (Salac, 2020) the curriculum *Comprehending Code* was created to utilize reading comprehension strategies and research to drive computer science curriculum. Just as reading requires strategies beyond decoding the letters into words, students need to make "meaning of the sequences of words into instructions (like sentences) and the sequences of instructions into functions or programs (like paragraphs)." However, we were unable to identify empirically based studies on whether these curricula were effective in increasing students' CT or literacy skills.

Although there are quite a few examples of CS and literacy integration, there are a limited number of empirical studies investigating the effectiveness of these integrations on literacy. Most of the literacy/CT studies are focused on students' CT skills improvements. For example, Lee (2010) taught and examined the CT knowledge and experiences of a one nine-year-old boy. Lee met with the student once-per week for six weeks and showed him basic programming and Scratch functions. Over the next 18 weeks, the boy selected and built language arts themed projects such as digital storybooks and games. Lee found that the visual programming approach employed in Scratch and the analogy-based instructional strategy enabled the young participant to successfully learn computer programming while creating a variety of multimedia products. Another study suggested that Scratch can be beneficial for non-native English speakers to learn the English. In a study of 32 4th and 5th graders in Spain, all students reported that this experience increased their English capabilities (Moreno-León & Robles, 2015). However, these were self-reported claims by the students.

Many of these studies and resources show the potential for integrating CS into literacy instruction (or the other way around). We have personally seen the power of building CS elementary instruction around stories and literature. There have been a range of different children's books focusing on computer science within the past few years. Specifically, these books tend to focus on using literacy to teach CS/CT ideas and principles. Haroldson and Bellard

(2020) reviewed 45 picture books and graphic novels published between 2015 and 2019 that focused on CS at the K-8 grade levels (see Figure 18 for a screenshot of the list captured from the article).

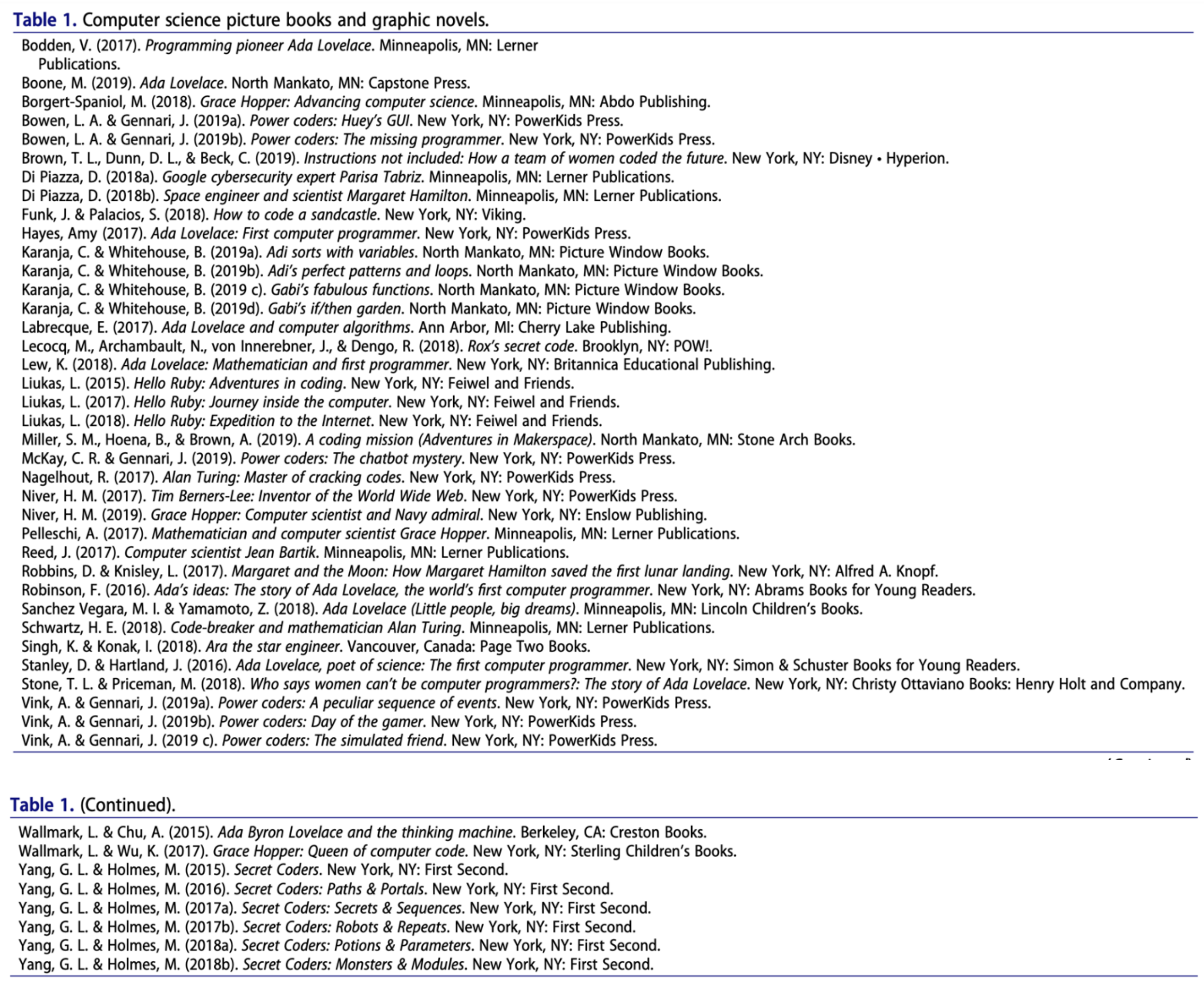


Table 1. Computer science picture books and graphic novels.

Bodden, V. (2017). *Programming pioneer Ada Lovelace*. Minneapolis, MN: Lerner Publications.
Boone, M. (2019). *Ada Lovelace*. North Mankato, MN: Capstone Press.
Borgert-Spaniol, M. (2018). *Grace Hopper: Advancing computer science*. Minneapolis, MN: Abdo Publishing.
Bowen, L. A. & Gennari, J. (2019a). *Power coders: Huey's GUI*. New York, NY: PowerKids Press.
Bowen, L. A. & Gennari, J. (2019b). *Power coders: The missing programmer*. New York, NY: PowerKids Press.
Brown, T. L., Dunn, D. L., & Beck, C. (2019). *Instructions not included: How a team of women coded the future*. New York, NY: Disney • Hyperion.
Di Piazza, D. (2018a). *Google cybersecurity expert Parisa Tabriz*. Minneapolis, MN: Lerner Publications.
Di Piazza, D. (2018b). *Space engineer and scientist Margaret Hamilton*. Minneapolis, MN: Lerner Publications.
Funk, J. & Palacios, S. (2018). *How to code a sandcastle*. New York, NY: Viking.
Hayes, Amy (2017). *Ada Lovelace: First computer programmer*. New York, NY: PowerKids Press.
Karanja, C. & Whitehouse, B. (2019a). *Adi sorts with variables*. North Mankato, MN: Picture Window Books.
Karanja, C. & Whitehouse, B. (2019b). *Adi's perfect patterns and loops*. North Mankato, MN: Picture Window Books.
Karanja, C. & Whitehouse, B. (2019 c). *Gabi's fabulous functions*. North Mankato, MN: Picture Window Books.
Karanja, C. & Whitehouse, B. (2019d). *Gabi's if/then garden*. North Mankato, MN: Picture Window Books.
Labrecque, E. (2017). *Ada Lovelace and computer algorithms*. Ann Arbor, MI: Cherry Lake Publishing.
Lecocq, M., Archambault, N., von Innerebner, J., & Dengo, R. (2018). *Rox's secret code*. Brooklyn, NY: POW!.
Lew, K. (2018). *Ada Lovelace: Mathematician and first programmer*. New York, NY: Britannica Educational Publishing.
Liukas, L. (2015). *Hello Ruby: Adventures in coding*. New York, NY: Feiwel and Friends.
Liukas, L. (2017). *Hello Ruby: Journey inside the computer*. New York, NY: Feiwel and Friends.
Liukas, L. (2018). *Hello Ruby: Expedition to the Internet*. New York, NY: Feiwel and Friends.
Miller, S. M., Hoena, B., & Brown, A. (2019). *A coding mission (Adventures in Makerspace)*. North Mankato, MN: Stone Arch Books.
McKay, C. R. & Gennari, J. (2019). *Power coders: The chatbot mystery*. New York, NY: PowerKids Press.
Nagelhout, R. (2017). *Alan Turing: Master of cracking codes*. New York, NY: PowerKids Press.
Niver, H. M. (2017). *Tim Berners-Lee: Inventor of the World Wide Web*. New York, NY: PowerKids Press.
Niver, H. M. (2019). *Grace Hopper: Computer scientist and Navy admiral*. New York, NY: Enslow Publishing.
Pelleschi, A. (2017). *Mathematician and computer scientist Grace Hopper*. Minneapolis, MN: Lerner Publications.
Reed, J. (2017). *Computer scientist Jean Bartik*. Minneapolis, MN: Lerner Publications.
Robbins, D. & Knisley, L. (2017). *Margaret and the Moon: How Margaret Hamilton saved the first lunar landing*. New York, NY: Alfred A. Knopf.
Robinson, F. (2016). *Ada's ideas: The story of Ada Lovelace, the world's first computer programmer*. New York, NY: Abrams Books for Young Readers.
Sanchez Vegara, M. I. & Yamamoto, Z. (2018). *Ada Lovelace (Little people, big dreams)*. Minneapolis, MN: Lincoln Children's Books.
Schwartz, H. E. (2018). *Code-breaker and mathematician Alan Turing*. Minneapolis, MN: Lerner Publications.
Singh, K. & Konak, I. (2018). *Ara the star engineer*. Vancouver, Canada: Page Two Books.
Stanley, D. & Hartland, J. (2016). *Ada Lovelace, poet of science: The first computer programmer*. New York, NY: Simon & Schuster Books for Young Readers.
Stone, T. L. & Priceman, M. (2018). *Who says women can't be computer programmers?: The story of Ada Lovelace*. New York, NY: Christy Ottaviano Books: Henry Holt and Company.
Vink, A. & Gennari, J. (2019a). *Power coders: A peculiar sequence of events*. New York, NY: PowerKids Press.
Vink, A. & Gennari, J. (2019b). *Power coders: Day of the gamer*. New York, NY: PowerKids Press.
Vink, A. & Gennari, J. (2019 c). *Power coders: The simulated friend*. New York, NY: PowerKids Press.

Table 1. (Continued).

Wallmark, L. & Chu, A. (2015). *Ada Byron Lovelace and the thinking machine*. Berkeley, CA: Creston Books.
Wallmark, L. & Wu, K. (2017). *Grace Hopper: Queen of computer code*. New York, NY: Sterling Children's Books.
Yang, G. L. & Holmes, M. (2015). *Secret Coders*. New York, NY: First Second.
Yang, G. L. & Holmes, M. (2016). *Secret Coders: Paths & Portals*. New York, NY: First Second.
Yang, G. L. & Holmes, M. (2017a). *Secret Coders: Secrets & Sequences*. New York, NY: First Second.
Yang, G. L. & Holmes, M. (2017b). *Secret Coders: Robots & Repeats*. New York, NY: First Second.
Yang, G. L. & Holmes, M. (2018a). *Secret Coders: Potions & Parameters*. New York, NY: First Second.
Yang, G. L. & Holmes, M. (2018b). *Secret Coders: Monsters & Modules*. New York, NY: First Second.

Figure 18. List of 45 Books Reviewed by Haroldson and Bellard (2020)

In other literary lists, CSTA's K-8 CS Integration Resources curated by Todd Lash and Vicky Sedgwick compiled resources and books that directly relate to CS/CT. The Canon Lab at the University of Chicago run by Diana Franklin also established a list of books that relate to CT appropriate concepts for K-5 students (https://www.canonlab.org/prekreadinglist). These include *If you give a mouse a cookie* for infinite loops, or *Beautiful oops* for persistence and the design process, or *The art of clean up* to discuss how to organize data. For each book, they have suggestions for how to incorporate ideas around CT, including summaries, key questions, strategies while reading the book and even coinciding activities.

One of the more popular books is targeted towards the lower elementary grade levels is *Hello Ruby*. This book series has three different books, each focusing on a different CS-related concepts. The first book focuses heavily on CT. It centers around a curious little girl, Ruby, who uses her imagination to embark on a journey to crack the code of a mysterious card left by her father before heading to work. Throughout this journey, Ruby needs to apply CT skills to abstract and identify patterns, implement loops/conditional statements, and construct solutions. She is a children's book character created by Linda Liukas as a role model for children to get immersed in the world of technology, computing and coding in a fun and playful, more inquiry-based way (Kruskopf, 2016). The CS in SF curriculum at the K-2 levels are heavily based on these books, incorporating the unplugged activities to explore basic control structures like sequencing, loops, and conditionals.

Integration of CT/CS into Elementary Music

Researchers are also investigating ways to incorporate CS into music and the arts at the primary school level. For example, Baratè et al. (2017) used LEGO bricks to represent basic musical notations and to show how pitch and time can be represented graphically (see Figure 19 below). Barate described that this approach requires ideas around abstraction, iteration, and debugging.

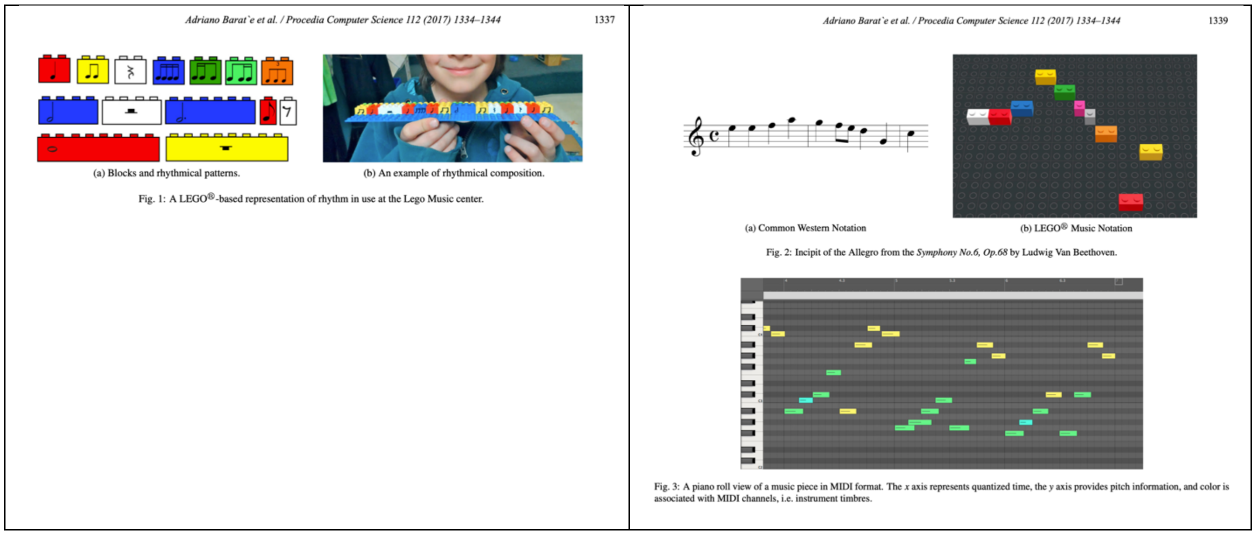

Adriano Barat`e et al. / Procedia Computer Science 112 (2017) 1334–1344
1337
(a) Blocks and rhythmical patterns.
(b) An example of rhythmical composition.
Adriano Barat`e et al. / Procedia Computer Science 112 (2017) 1334–1344
1339
(a) Common Western Notation
(b) LEGO® Music Notation
Fig. 1: A LEGO®-based representation of rhythm in use at the Lego Music center.

Fig. 2: Incipit of the Allegro from the *Symphony No.6, Op.68* by Ludwig Van Beethoven.

Fig. 3: A piano roll view of a music piece in MIDI format. The *x* axis represents quantized time, the *y* axis provides pitch information, and color is associated with MIDI channels, i.e. instrument timbres.

Figure 19. Screenshots from Baratè et al. (2020)

Another example of incorporating music and coding is through Wonder Workshop's Dash, a robot that can produce sounds and light depending on input from stimuli and/or coding inputted through an app. WonderWorkshop has designed lesson plans to introduce students to music and coding at the Kindergarten and first grade levels. Students are able to visualize the notes in a hierarchical order and over time. They can program the robot to play the notes on the xylophone through a simple app (see Figure 20).

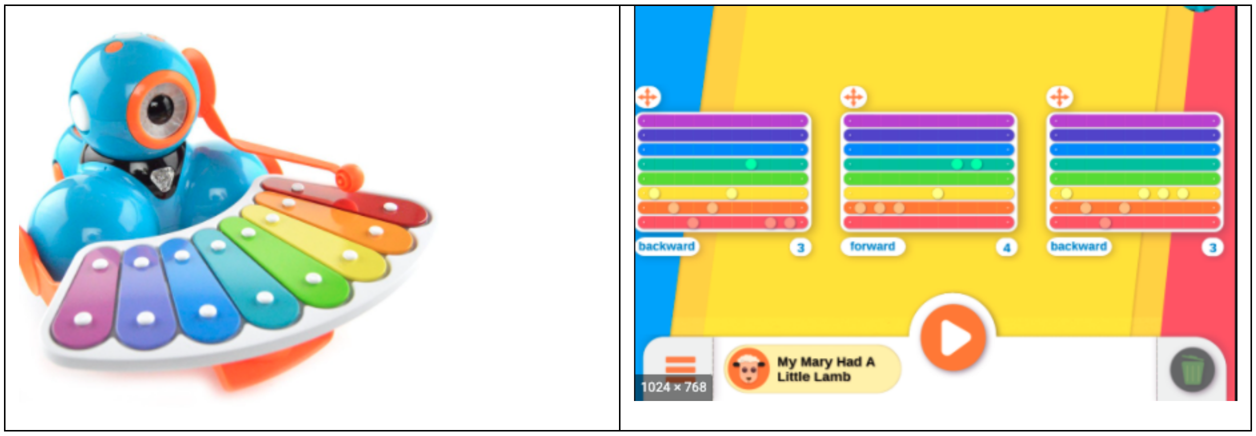


Figure 20. Screenshot examples from WonderWorkshop's website.

Another more advanced robot for integrating CT into music is Wigl (http://wiglbot.com/). This robot responds to specific notes with real-world movements and lights. For example, playing an "A" commands the robot to move forward, "B" backwards, "C" turns left, and "D" turns right. It hears notes (from any instrument and even singing) and responds with real-world movements, lights, and special dances. Through sequenced musical notes, you can even program unique moves. Another interesting initiative is *Note Code*, which is a music programming puzzle game designed as a tangible device coupled with entities to store sets of notes, play them back and activate different sub-components or neighboring boxes.

Google's CS First curriculum also has a unit dedicated to music and sound consisting of eight different activities. In addition, Vicky Sedgwick has created curricular integration ideas using the Micro:Bit with music.

Integration of CT/CS into Elementary Arts

In terms of the arts, ScratchJr and Scratch both enable students to create their own characters and backgrounds. There was little research on this at the K-5 levels, although other research around e-Textiles and the arts exists at the secondary level (Lui et al., 2020). At the elementary level, Scratch's Cross-Curricular Integration Guide included several examples and resources on how to integrate Scratch and fine arts. Some of the 3rd - 8th grade project examples include colors of the rainbow, making interactive landscapes, and animating aesop fables. Google's CS First curriculum also includes an intermediate art unit with eight activities that leads students through the creation of animation, interactive artwork, photograph filters and other exciting artistic projects using code.

Challenges with Integration of CT/CS into Elementary Classrooms

Teachers and scholars reported many challenges in teaching computational thinking/computer science. Some of these challenges include frequent policy changes (B. Barker, 2010), mandatory requirements from district and schools (e.g., Google Inc. & Gallup Inc, 2016; Indiana Department of Education, 2018), and even demands from the parents to teach CS/CT (Google Inc. & Gallup Inc, 2016; Wang et al., 2016). Perhaps the largest concern is due to competing demands from other subject areas and testing, teachers often struggle with addressing CT/CS in their curriculum. Scholars have suggested integrating CT/CS into existing curriculum (Barr & Stephenson, 2011), especially through shared terminology like those found in math (Barr & Stephenson, 2011; Rich et al., 2019; Sneider et al., 2014; Weintrop et al., 2016). Or perhaps integrating CT/CS by building content connections (Sung et al., 2017). Or through the potential of building pedagogy connections through elementary problem-based learning (Ozturk et al., 2018).

However, teachers still describe challenges with this approach due to their lack of content knowledge around CT/CS. Teachers have often reported receiving a lack of training in CS/CT at the elementary level (e.g., Ozturk et al., 2018), and often indicate feeling underprepared and unable to incorporate these ideas into existing curriculum (Ottenbreit-Leftwich & Biggers, 2017). Israel et al. (2020) and Ray et al. (2018) both found that teachers were challenged to differentiate elementary CS instruction for varying levels of academic abilities. They both noted that teachers required the support of an instructional coach (with CS and UDL background) as well as significant PD to successfully achieve differentiated CS instruction for all learners.

Experts have argued that in order for K-12 teachers to be able to integrate CT, they must have professional support (Yadav, Gretter, et al., 2017). It appears that providing support to teachers in the form of curriculum, coaching, or professional development is critical to the success of incorporating CT into the existing K-5 curriculum. Furthermore, policies and daily support structures need to be established to create the system that enables CT integration. In other words, elementary teachers need to know and feel the importance of CT/CS. This can be easily addressed by school administration acknowledging the importance of CT and providing teachers with the support to be able to integrate CT into their classrooms.

Conclusion

Computational thinking (CT) in PreK-5 is an emerging and developing field. There is still a great deal that we must learn about the developmental appropriateness of different techniques for teaching CT, especially at the elementary level. Learning trajectories and progressions are currently being developed and studied, but much of the research is not yet published. There are a limited number of curricular innovations that are ready for PreK-5 implementation, and even fewer that have been empirically tested, and even fewer still that are integrated into other subject areas with attention to meaningful content integration between subjects.

The rationale for integration is two-fold. First, many schools face great challenges to add CT into their curriculum. One of the greatest challenges is the time dedicated to other subject areas. Second, studies have shown that contextualized application of CT tend to make learning more relevant and meaningful for students. Therefore, curricula that uses CT to also teach or reinforce other content areas through an integrated approach will be valuable for K-5 schools. CT integration at the K-5 levels is most commonly being done with engineering design-based STEM integration, with mathematics, and with literacy. The connections between engineering and CT are natural and have a lot of room for development. The ways of thinking in mathematics and CT are very similar when we think about it from a conceptual point of view. Using CT to reinforce literacy and reading is being shown to have great potential. These areas need more curricular resources for all grade bands.

Furthermore, curricular innovations can fall in multiple areas: unplugged, plugged, tools, robotics, and books. Curricular resources that fall into each of these categories and integrate with other subjects that are required to be taught should be created to focus on (1) clear, developmentally appropriate learning trajectories and strategies, (2) using multiple representations, and (3) making CT learning active, hands-on, and minds-on. Two stand out curricular examples that incorporate these elements were PictureSTEM and ETA Hand2Mind: STEM in Action.

We suggest that future research continues to investigate the connections between the disciplines to more fully integrate CT in meaningful ways that do not add to the curriculum for students, but rather enhances it for all areas. Curricula need to be developed and studied to address the concerns brought up here. There is also a need for a deeper understanding of how students develop CT competencies both with and without integration. Very few clinical studies exist of how students engage with CT at different developmental stages or with a mind to the CT skills that need to be developed. New studies that look at CT comprehensively and breakdown the CT skills to focus deeply on these are needed. These studies should be used as a basis for developing learning opportunities for children in both formal and informal spaces.

REFERENCES

Aggarwal, A., Gardner-McCune, C., & Tourestzky, D. (2017). Evaluating the effect of using physical manipulatives to foster computational thinking in elementary school. ACM SIGCSE Technical Symposium on Computer Science Education, Seattle, Washington.

Angeli, C., Voogt, J., Fluck, A., Webb, M., Cox, M., Malyn-Smith, J., & Zagami, J. (2016). A K-6 computational thinking curriculum framework: Implications for teacher knowledge. *Journal of Educational Technology & Society, 19*(3), 47-57.

Balanskat, A. and Engelhardt, K. (2014): Computing our future: computer programming and coding - priorities, school curricula, and initiatives across Europe. European Schoolnet. Available at: http://www.eun.org/c/document_library/get_file?uuid=521cb928-6ec4-4a86-b522- 9d8fd5cf60ce&groupId=43887

Baratè, A., Ludovico, L. A., & Malchiodi, D. (2017). Fostering computational thinking in primary school through a LEGO®-based music notation. *Procedia computer science, 112*, 1334-1344.

Barker, B. (2010). *The pendulum swings: Transforming school reform*. Westview House 734 London Road, Oakhill, Stoke-on-Trent, Staffordshire, ST4 5NP, UK.: Trentham Books Ltd. .

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.

Berland, L. K., & Steingut, R. (2016). Explaining variation in student efforts towards using math and science knowledge in engineering contexts. *International Journal of Science Education, 38*(18), 2742-2761. http://doi.org/10.1080/09500693.2016.1260179

Bers, M. U. (2010). The tangible K robotics program: Applied computational thinking for young children. *Early Childhood Research and Practice, 12*(2). https://ecrp.illinois.edu/v12n2/bers.html

Bers, M. U. (2018a). Coding and computational thinking in early childhood: the impact of ScratchJr in Europe. *European Journal of STEM Education, 3*(3), 8.

Bers, M. U. (2018b). *Coding as a playground: Programming and computational thinking in the early classroom*. Routledge.

Breiner, J. M., Harkness, S. S., Johnson, C. C., & Koehler, C. M. (2012). What is STEM? A discussion about conceptions of STEM in education and partnerships. *School Science and Mathematics, 112*(1), 3-11. https://doi.org/10.1111/j.1949-8594.2011.00109.x

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. Proceedings of the 2012 annual meeting of the American educational research association, Vancouver, Canada,

Brown, N. C., Sentance, S., Crick, T., & Humphreys, S. (2014). Restart: The resurgence of computer science in UK schools. *ACM Transactions on Computing Education (TOCE), 14*(2), 1-22.

Bruner, J. S. (1960). *The process of education*. Harvard University Press.

Buitrago Flórez, F., Casallas, R., Hernández, M., Reyes, A., Restrepo, S., & Danies, G. (2017). Changing a generation's way of thinking: Teaching computational thinking through programming. *Review of Educational Research, 87*(4), 834-860.

Burke, Q., & Kafai, Y. B. (2012, February). 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* (pp. 433-438).

Byers, J. A., & Walker, C. (1995). Refining the motor training hypothesis for the evolution of play. *The American Naturalist, 146*(1), 25-40. https://doi.org/10.1086/285785

Century, J., Ferris, K. A., & Zuo, H. (2020). Finding time for computer science in the elementary school day: a quasi-experimental study of a transdisciplinary problem-based learning approach. *International Journal of STEM Education, 7*, 1-16.

Chalmers, C. (2018). Robotics and computational thinking in primary school. *International Journal of Child-Computer Interaction, 17*, 93-100.

Clark, J., Rogers, M. P., Spradling, C., & Pais, J. (2013). What, no canoes? Lessons learned while hosting a scratch summer camp. *Journal of Computing Sciences in Colleges, 28*, 204-210.

Corlu, M. A., & Aydin, E. (2016). Evaluation of learning gains through integrated STEM projects. *International Journal of Education in Mathematics, Science and Technology, 4*(1), 20-29. https://doi.org/10.18404/ijemst.35021

Curzon, P., Bell, T., Waite, J., & Dorling, M. (2018). Computational thinking. In S. A. Fincher & A. V. Robins (Eds.), *The Cambridge Handbook of Computing Education Research* (pp. 513-546). Cambridge University Press.

del Olmo-Muñoz, J., Cózar-Gutiérrez, R., & González-Calero, J. A. (2020). Computational thinking through unplugged activities in early years of primary education. *Computers & Education, 150*, 103832.

Denning, P. J., Tedre, M., & Yongpradit, P. (2017). The profession of IT misconceptions about computer science. *Communications of the ACM 60*(3), 31-33. http://hdl.handle.net/10945/60896

Dickes, A. C., Farris, A. V., & Sengupta, P. (2020). Sociomathematical norms for integrating coding and modeling with elementary science: A dialogical approach. *Journal of Science Education and Technology, 29*(1), 35-52. https://doi.org/10.1007/s10956-019-09795-7

Ehsan, H., Cardella, M. E. & Hynes, M. (2020, Apr 17 - 21) *Exploring Computational Thinking Engagement: An Exploratory Study on Children With Mild Autism*. AERA Annual Meeting San Francisco, CA. (Conference Canceled).

Ehsan, H., Cardella, M., & Svarovsky, G. (2018, April) Engineering and computational thinking among families engaging with an exhibit. Paper presented at the *American Educational Research Association (AERA) Annual Meeting*. New York City. NY..

Ehsan, H., Ohland, C., Cardella, M. (2018, June). Computing for the critters: Exploring computational thinking of children in informal learning settings. IEEE Frontiers in Education Conference, San Jose, CA. https://doi.org/10.1109/FIE.2018.8659268

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*. https://doi.org/10.1007/s10798-020-09562-5

Ehsan, H., Rehmat, A., & Cardella, M. E. (2019). Computer science unplugged: Design a puppy playground using computational thinking. *NSTA Science and Children, 57*(3), 32-38.

Faber, H. H., Wierdsma, M. D., Doornbos, R. P., van der Ven, J. S., & de Vette, K. (2017). Teaching computational thinking to primary school students via unplugged programming lessons. *Journal of the European Teacher Education Network, 12*, 13-24.

Fagundes, B., Ehsan, H., Moore, T. J., Tank, K. M., & Cardella, M. E. (2020, June). WIP: First-graders' computational thinking in informal learning settings. ASEE Virtual Annual Conference. https://doi.org/10.18260/1-2--35541

Falkner, K., Vivian, R., & Falkner, N. (2014, January). The Australian digital technologies curriculum: challenge and opportunity. In *Proceedings of the Sixteenth Australasian Computing Education Conference-Volume 148* (pp. 3-12).

Fessakis, G., Gouli, E., & Mavroudi, E. (2013, 2013/04/01/). Problem solving by 5–6 years old kindergarten children in a computer programming environment: A case study. *Computers & Education, 63*, 87-97. https://doi.org/https://doi.org/10.1016/j.compedu.2012.11.016

Flannery, L. P., Silverman, B., Kazakoff, E. R., Bers, M. U., Bontá, P., & Resnick, M. (2013). *Designing ScratchJr: support for early childhood learning through computer programming* Proceedings of the 12th International Conference on Interaction Design and Children, New York, New York, USA. https://doi.org/10.1145/2485760.2485785

Fluck, A. E., Webb, M., Cox, M. J., Angeli, C., Malyn-Smith, J., Voogt, J., & Zagami, J. (2016). Arguing for computer science in the school curriculum. *Educational Technology & Society, 19*(3), 38-46.

Google Inc. & Gallup Inc. (2016). Trends in the State of Computer Science in U.S. K-12 Schools. Retrieved from http://goo.gl/j291E0.

Gretter, S., & Yadav, A. (2016). Computational thinking and media & information literacy: An integrated approach to teaching twenty-first century skills. *TechTrends, 60*(5), 510-516.

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

Grover, S., Pea, R., & Cooper, S. (2016). Factors influencing computer science learning in middle school. Proceedings of the 47th ACM technical symposium on computing science education, (pp. 552-557).

Guo, M., & Ottenbreit-Leftwich, A. (2020). Exploring the k-12 computer science curriculum standards in the U.S. Workshop in Primary and Secondary Computing Education, Online.

Guzey, S. S., Moore, T. J., & Harwell, M. (2016). Building up STEM: An analysis of teacher-developed engineering design-based STEM integration curricular materials. *Journal of Pre-College Engineering Education Research (J-PEER), 6*(1), 11-29. https://doi.org/https://doi.org/10.7771/2157-9288.1129

Haroldson, R., & Ballard, D. (2020). Alignment and representation in computer science: an analysis of picture books and graphic novels for K-8 students. *Computer Science Education,* 1-26.

Heintz, F., Mannila, L., & Farnqvist, T. (2016, October). A review of models for introducing computational thinking, computer science and computing in K-12 education. *IEEE Frontiers in Education Conference*, Erie, PA. https://doi.org/10.1109/FIE.2016.7757410

Henderson, P. B., Cortina, T. J., & Wing, J. (2007). *Computational thinking* Proceedings of the 38th SIGCSE technical symposium on Computer science education, Covington, Kentucky, USA. https://doi.org/10.1145/1227310.1227378

Hermans, F., & Aivaloglou, E. (2017). To scratch or not to scratch? A controlled experiment comparing plugged first and unplugged first programming lessons. Proceedings of the 12th workshop on primary and secondary computing education,

Hsiao, H.-S., Lin, Y.-W., Lin, K.-Y., Lin, C.-Y., Chen, J.-H., & Chen, J.-C. (2019). Using robot-based practices to develop an activity that incorporated the 6E model to improve elementary school students' learning performances. *Interactive Learning Environments*, 1-15. https://doi.org/10.1080/10494820.2019.1636090

Hynes, M. M., Cardella, M. E., Moore, T. J., Brophy, S. P., Purzer, S., Tank, K. M., Menekse, M., Yeter, I. H., & Ehsan, H. (2019, June). Inspiring young children to engage in computational thinking in and out of school (Research-to-practice). American Society for Engineering Education Annual Conference & Exposition, Tampa, FL.

Indiana Department of Education. (2018b). *STEM Six-Year strategic Plan: An Integrated K-12 STEM Approach for Indiana* Retrieved from https://www.doe.in.gov/sites/default/files/wf-stem/20181108154535030.pdf

Israel, M., Jeong, G., Ray, M., & Lash, T. (2020). *Teaching Elementary Computer Science through Universal Design for Learning* Proceedings of the 51st ACM Technical Symposium on Computer Science Education, Portland, OR, USA. https://doi-org.proxyiub.uits.iu.edu/10.1145/3328778.3366823

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 & Education, 82*, 263-279.

Jacob, S. R., & Warschauer, M. (2018). Computational thinking and literacy. *Journal of Computer Science Integration, 1*(1).

Johnson, C. C., Peters-Burton, E. E., & Moore, T. J. (2016). *STEM road map: A framework for integrated STEM education* [Book]. Routledge. http://search.ebscohost.com/login.aspx?direct=true&db=nlebk&AN=1021248&site=ehost-live

K–12 Computer Science Framework. (2016). http://www.k12cs.org

Kelleher, C. & Pausch, R., & Kiesler, S. (2007). Storytelling Alice motivates middle school girls to learn computer programming. *Conference on Human Factors in Computing Systems - Proceeding*s. P. 1455-1464. 10.1145/1240624.1240844.

Kruskopf, M. (2016). Explorations on the nature of children's conceptual change in computational thinking during hello ruby summer school 2016. Master's Thesis, University of Helsinki. Retrieved from https://helda.helsinki.fi/bitstream/handle/10138/174410/MastersThesisMillaKruskopf2016.pdf?isAllowed=y&sequence=2

Kalelioğlu, F. (2015). A new way of teaching programming skills to K-12 students: Code. org. *Computers in Human Behavior, 52*, 200-210.

Kalelioğlu, F. (2015). A new way of teaching programming skills to K-12 students: Code. org. *Computers in Human Behavior, 52*, 200-210.

Kong, S.-C., Chiu, M. M., & Lai, M. (2018). A study of primary school students' interest, collaboration attitude, and programming empowerment in computational thinking education. *Computers & Education, 127*, 178-189.

Lambić, D., Đorić, B., & Ivakić, S. (2020). Investigating the effect of the use of code. org on younger elementary school students' attitudes towards programming. *Behaviour & Information Technology*, 1-12.

Lambić, D., Đorić, B., & Ivakić, S. (2020). Investigating the effect of the use of code. org on younger elementary school students' attitudes towards programming. *Behaviour & Information Technology*, 1-12.

Lavigne, H. J., Lewis-Presser, A., & Rosenfeld, D. (2020). An exploratory approach for investigating the integration of computational thinking and mathematics for preschool children. *Journal of Digital Learning in Teacher Education, 36*(1), 63-77.

Lee, Y.-J. (2010). Developing computer programming concepts and skills via technology-enriched language-art projects: A case study. *Journal of Educational Multimedia and Hypermedia, 19*(3), 307-326.

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

Lesh, R., & Doerr, H. M. (2003). Foundations of a models and modeling perspective on mathematics teaching, learning, and problem solving. In R. Lesh & H. M. Doerr (Eds.), *Beyond constructivism: Models and modeling perspectives on mathematics problem solving, learning, and teaching* (pp. 3-34). Lawrence Erlbaum.

Lesh, R., & Harel, G. (2003). Problem solving, modeling, and local conceptual development. *Mathematical Thinking and Learning, 5*(2), 157-189. https://doi.org/https://doi.org/10.1207/S15327833MTL0502&3_03

Lewis, C. M., & Shah, N. (2012, February). Building upon and enriching grade four mathematics standards with programming curriculum. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education* (pp. 57-62).

Lowe, T., & Brophy, S. (2019). Identifying computational thinking in storytelling literacy activities with Scratch Jr. In *ASEE Annual Conference Proceeding*s (p. 10).

Lui, D., Walker, J. T., Hanna, S., Kafai, Y. B., Fields, D., & Jayathirtha, G. (2020). Communicating computational concepts and practices within high school students' portfolios of making electronic textiles. *Interactive Learning Environments*, *28*(3), 284-301.

Lye, S. Y., & Koh, J. H. L. (2014, 2014/12/01/). Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in Human Behavior, 41*, 51-61. https://doi.org/10.1016/j.chb.2014.09.012

Malik, S. I., Shakir, M., Eldow, A., & Ashfaque, M. W. (2019). Promoting algorithmic thinking in an introductory programming course. *International Journal of Emerging Technologies in Learning, 14*(1), 84-94. https://doi.org/10.3991/ijet.v14i01.9061

Maloney, J., Resnick, M., Rusk, N., Silverman, B., & Eastmond, E. (2010). The scratch programming language and environment. *ACM Transactions on Computing Education (TOCE), 10*(4), 1-15.

Milesi, C., Perez-Felkner, L., Brown, K., & Schneider, B. (2017, 2017-April-25). Engagement, Persistence, and Gender in Computer Science: Results of a Smartphone ESM Study [Original Research]. *Frontiers in Psychology, 8*(602). https://doi.org/10.3389/fpsyg.2017.00602

Moore, T. J., Brophy, S. P., Tank, K. M., Lopez, R. D., Johnston, A. C., Hynes, M. M., & Gajdzik, E. (2020). Multiple representations in computational thinking tasks: A clinical study of second-grade students. *Journal of Science Education and Technology, 29*(1), 19-34. https://doi.org/10.1007/s10956-020-09812-0

Moore, T. J., Johnston, A. C., & Glancy, A. W. (2020). STEM integration: A synthesis of conceptual frameworks and definitions. In C. C. Johnson, M. J. Mohr-Schroeder, T. J. Moore, & L. D. English (Eds.), *Handbook of Research on STEM Education* (pp. 3-16). Routledge.

Moreno-León, J., & Robles, G. (2015, March). Computer programming as an educational tool in the English classroom a preliminary study. In *2015 IEEE Global Engineering Education Conference (EDUCON)* (pp. 961-966). IEEE.

Mossberger, K. (2009). Toward digital citizenship. Addressing inequality in the information age. *Routledge handbook of Internet politics, 173*, 85.

Mossberger, K., Tolbert, C. J., & McNeal, R. S. (2007). *Digital citizenship: The Internet, society, and participation*. MIT Press.

Mouza, C., Yadav, A., & Ottenbreit-Leftwich, A. (2018). Developing computationally literate teachers: Current perspectives and future directions for teacher preparation in computing education. *Journal of Technology and Teacher Education, 26*(3), 333-352.

Nathan, M. J., Srisurichan, R., Walkington, C., Wolfgram, M., Williams, C., & Alibali, M. W. (2013). Building cohesion across representations: A mechanism for STEM integration. *Journal of Engineering Education, 102*(1), 77-116. https://doi.org/http://doi.org/10.1002/jee.20000

National Research Council. (2010). *Report of a workshop on the scope and nature of computational thinking*. The National Academies Press. https://doi.org/10.17226/12840

National Research Council. (2011). *Report of a workshop on the pedagogical aspects of computational thinking.* The National Academies Press. https://doi.org/10.17226/13170

Nouri, J., Zhang, L., Mannila, L., & Norén, E. (2020). Development of computational thinking, digital competence and 21st century skills when learning programming in K-9. *Education Inquiry, 11*(1), 1-17.

Ottenbreit-Leftwich, A.T. & Biggers, M. (2017). Status of K-14 computer science education in Indiana: Landscape Report. Submitted to the NSF's ECEP Alliance, the Indiana Department of Education, Governor of Indiana, Code.org, and Indiana legislators. http://bit.ly/CSforINFinalReport

Ozturk, Z., Dooley, C. M., & Welch, M. (2018). Finding the hook: Computer science education in elementary contexts. *Journal of Research on Technology in Education, 50*(2), 149-163.

Papadakis, S., Kalogiannakis, M., & Zaranis, N. (2016). Developing fundamental programming concepts and computational thinking with ScratchJr in preschool education: a case study. *International Journal of Mobile Learning and Organisation, 10*(3), 187-202.

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

Pila, S., Aladé, F., Sheehan, K. J., Lauricella, A. R., & Wartella, E. A. (2019). Learning to code via tablet applications: An evaluation of Daisy the Dinosaur and Kodable as learning tools for young children. *Computers & Education, 128*, 52-62.

Pila, S., Aladé, F., Sheehan, K. J., Lauricella, A. R., & Wartella, E. A. (2019). Learning to code via tablet applications: An evaluation of Daisy the Dinosaur and Kodable as learning tools for young children. *Computers & Education, 128*, 52-62.

Ray, M. J., Israel, M., Lee, C., & Do, V. (2018). *A Cross-Case Analysis of Instructional Strategies to Support Participation of K-8 Students with Disabilities in CS for All* Proceedings of the 49th ACM Technical Symposium on Computer Science Education, Baltimore, Maryland, USA. https://doi.org/10.1145/3159450.3159482

Ribble, M. (2015). *Digital citizenship in schools: Nine elements all students should know* (third ed.). International Society for Technology in Education.

Rich, K. M., Binkowski, T. A., Strickland, C., & Franklin, D. (2018). *Decomposition: A K-8 computational thinking learning trajectory* Proceedings of the 2018 ACM Conference on International Computing Education Research, Espoo, Finland. https://doi-org.proxyiub.uits.iu.edu/10.1145/3230977.3230979

Rich, K. M., Spaepen, E., Strickland, C., & Moran, C. (2020). Synergies and differences in mathematical and computational thinking: Implications for integrated instruction. *Interactive Learning Environments, 28*(3), 272-283. https://doi.org/10.1080/10494820.2019.1612445

Rich, K. M., Yadav, A., & Schwarz, C. V. (2019). Computational thinking, mathematics, and science: Elementary teachers' perspectives on integration. *Journal of Technology and Teacher Education, 27*(2), 165-205.

Roosevelt, E. (2008). Good citizenship: The purpose of education. *Yearbook of the National Society for the Study of Education, 107*(2), 312-320.

Ryan, M., Gale, J., & Usselman, M. (2017). Integrating engineering into core science instruction: Translating NGSS principles into practice through iterative curriculum design. *International Journal of Engineering Education, 33*(1B), 321-331.

Ryu, M., Mentzer, N., & Knobloch, N. (2018). Preservice teachers' experiences of STEM integration: Challenges and implications for integrated STEM teacher preparation. *International Journal of Technology and Design Education*. http://doi.org/10.1007/s10798-018-9440-9

Sáez-López, J.-M., Román-González, M., & Vázquez-Cano, E. (2016). Visual programming languages integrated across the curriculum in elementary school: A two year case study using "Scratch" in five schools. *Computers & Education, 97*, 129-141.

Sahami, M., Roach, S., Cuadros-Vargas, E., & LeBlanc, R. (2013). ACM/IEEE-CS computer science curriculum 2013: reviewing the ironman report. Proceeding of the 44th ACM technical symposium on Computer science education,

Salac, J., Thomas, C., Twarek, B., Marsland, W., & Franklin, D. (2020, February). Comprehending code: Understanding the relationship between reading and math proficiency, and 4th-grade cs learning outcomes. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education* (pp. 268-274).

Seiter, L., & Foreman, B. (2013). *Modeling the learning progressions of computational thinking of primary grade students* Proceedings of the ninth annual international ACM conference on International computing education research, San Diego, San California, USA. https://doi-org.proxyiub.uits.iu.edu/10.1145/2493394.2493403

Shein, E. (2014). Should everybody learn to code? *Communications of the ACM 57*(2), 16–18. DOI:https://doi.org/10.1145/2557447

Sneider, C., Stephenson, C., Schafer, B., & Flick, L. (2014). Exploring the science framework and NGSS: Computational thinking in the science classroom. *Science Scope, 38*(3), 10-15.

Stubbs, E. A., & Myers, B. E. (2015, 06//). Multiple case study of STEM in school-based agricultural education [Article]. *Journal of Agricultural Education, 56*(2), 188-203. https://doi.org/http://doi.org/10.5032/jae.2015.02188

Sullivan, A., & Bers, M. U. (2019). Vex robotics competitions: Gender differences in student attitudes and experiences. *Journal of Information Technology Education: Research, 18*, 97-112. https://doi.org/10.28945/4193

Sullivan, F. R., & Heffernan, J. (2016). Robotic construction kits as computational manipulatives for learning in the STEM disciplines. *Journal of Research on Technology in Education, 48*(2), 105-128. https://doi.org/10.1080/15391523.2016.1146563

Sung, W., Ahn, J., & Black, J. B. (2017). Introducing computational thinking to young learners: Practicing computational perspectives through embodiment in mathematics education. *Technology, Knowledge and Learning, 22*(3), 443-463.

Sysło, M. M., & Kwiatkowska, A. B. (2015). Introducing a new computer science curriculum for all school levels in Poland. In A. Brodnik & J. Vahrenhold (Eds.), *Informatics in schools. Curricula, competences, and competitions* (pp. 141-154). Springer International Publishing. https://doi.org/10.1007/978-3-319-25396-1_13

Tank, K. M., Moore, T. J., Dorie, B. L., Gajdzik, E., Terri Sanger, M., Rynearson, A. M., & Mann, E. F. (2018). Engineering early elementary classrooms through the integration of high-quality literature, design, and STEM+C content. In L. English & T. Moore (Eds.), *Early Engineering Learning* (pp. 175-201). Springer Singapore. https://doi.org/10.1007/978-981-10-8621-2_9

Touretzky, D. S., & Gardner-McCune, C. (2018). Calypso for Cozmo: Robotic AI for everyone. *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, Baltimore, Maryland, USA. https://doi.org/10.1145/3159450.3162200

Vogel, S., Santo, R., & Ching, D. (2017, March). 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* (pp. 609-614).

Vogel, S., Santo, R., & Ching, D. (2017, March). 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 (pp. 609-614).

Wang, J., Hong, H., Ravitz, J., & Hejazi Moghadam, S. (2016). Landscape of K-12 computer science education in the US: Perceptions, access, and barriers. Paper presented at the *Proceedings of the 47th ACM Technical Symposium on Computing Science Education.*

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.

Wilson, C. (2013). Making computer science count. *Communications of the ACM 56*(11), 32–33. DOI:https://doi.org/10.1145/2527189

Wing, J. M. (2006). Computational thinking. *Commun. ACM, 49*(3), 33–35. https://doi.org/10.1145/1118178.1118215

Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical transactions. Series A, Mathematical, physical, and engineering sciences, 366*(1881), 3717-3725. https://doi.org/10.1098/rsta.2008.0118

Wohl, B., Porter, B., & Clinch, S. (2015). Teaching Computer Science to 5-7 year-olds: An initial study with Scratch, Cubelets and unplugged computing. proceedings of the workshop in primary and secondary computing education,

Yadav, A., Good, J., Voogt, J., & Fisser, P. (2017). Computational thinking as an emerging competence domain. In *Competence-based vocational and professional education* (pp. 1051-1067). Springer.

Yadav, A., Gretter, S., Good, J., & McLean, T. (2017). Computational thinking in teacher education. In P. J. Rich & C. B. Hodges (Eds.), *Emerging Research, Practice, and Policy on Computational Thinking* (pp. 205-220). Springer International Publishing. https://doi.org/10.1007/978-3-319-52691-1_13

Yadav, A., Hong, H., & Stephenson, C. (2016). Computational thinking for all: Pedagogical approaches to embedding 21st century problem solving in K-12 classrooms. *TechTrends, 60*(6), 565-568.

Yadav, A., Larimore, R., Rich, K., & Schwarz, C. (2019). *Integrating computational thinking in elementary classrooms: Introducing a toolkit to support teachers* Society for Information Technology & Teacher Education International Conference 2019, Las Vegas, NV, United States. https://www.learntechlib.org/p/208366

Zhang, L., & Nouri, J. (2019). A systematic review of learning computational thinking through Scratch in K-9. *Computers & Education, 141*, 103607.

Appendix

The appendix for this white paper was developed by Meize Guo, a PhD candidate at Indiana University, and Barbara Fagundes, a PhD student at Purdue University. Ms. Guo is advised by Dr. Anne Ottenbreit-Leftwich, and Ms. Fagundes is advised by Dr. Tamara Moore. The contents of the Appendix can be found at: https://docs.google.com/spreadsheets/d/1eo4-ReBlKt6RaTQkFBZzv5BXGSuhb7wcxH7ezfzkpR8/edit?usp=sharing