Computational Thinking in PreK-5: Empirical Evidence for Integration and Future Directions

Edited by:
Dr. Anne Ottenbreit-Leftwich
Indiana University
Dr. Aman Yadav
Michigan State University

A Special Research Publication

Association for Computing Machinery

ROBIN HOOD LEARNING + TECHNOLOGY FUND
IN PARTNERSHIP WITH OVERDECK FAMILY FOUNDATION AND SIEGEL FAMILY ENDOWMENT

Computational Thinking in PreK-5: Empirical Evidence for Integration and Future Directions

Edited by:
Dr. Anne Ottenbreit-Leftwich
Indiana University
Dr. Aman Yadav
Michigan State University

A Special Research Publication

Association for Computing Machinery

ROBIN HOOD LEARNING + TECHNOLOGY FUND
IN PARTNERSHIP WITH OVERDECK FAMILY FOUNDATION AND SIEGEL FAMILY ENDOWMENT

Association for Computing Machinery
1601 Broadway, 10th Floor
New York, NY 10019-7434

ISBN: 978-1-4503-9615-8
DOI: 10.1145/3507951
Web link: https://dl.acm.org/citation.cfm?id=3507951

Recommended citation:

Ottenbreit-Leftwich, A., & Yadav, A. (2021). Computational Thinking in PreK-5: Empirical Evidence for Integration and Future Directions. ACM and the Robin Hood Learning + Technology Fund, New York, NY

Acknowledgements:

The authors thank ACM and the Robin Hood Learning + Technology Fund for supporting this publication. All opinions reflected in the papers in this publication are those of the authors and not necessarily those of ACM and the Robin Hood Learning + Technology Fund.

CONTENTS

INTRODUCTION

Computational Thinking in PreK-5: Empirical Evidence for Integration and Future Directions

Anne Ottenbreit-Leftwich, *Indiana University*
Aman Yadav, *Michigan State University*

Since Wing's (2006) article regarding computational thinking, there has been a dramatic increase in the focus on computational thinking (CT) in K-12 education. Computational thinking has permeated across K-12 classrooms, particularly at the elementary level. In a review of the state of the field of computational thinking in K-12, Grover and Pea (2013) described the research on CT as mainly focused on definitional issues, descriptions of environments and tools that foster CT, and assessment of CT. They called for a need to address large gaps in our knowledge around CT, including the cognitive aspects of CT and how CT can be integrated into other subjects. Several articles have provided suggestions on how to integrate CT (e.g., Yadav, Hong, & Stephenson, 2016). In a meta-review of 120 studies on CT published between 2006 to 2017, Hsu, Chang, and Hung (2018) found that most of these studies were situated within programming (n=31) or computer science (n=26) contexts. The other contexts where CT was studied included math (n=11), biology (n=9), and robotics (n=8). Many of these studies usually focused on students at the secondary and post-secondary levels.

Although research on CT within K-12 has been emerging over the past few years, few studies have investigated the teaching of CT at the younger ages. The ACM and the Robin Hood Learning + Technology Fund co-funded this special research publication to examine empirically-based studies that focused on the integration of computational thinking at the elementary levels into a variety of learning disciplines including math, ELA, science, and computer science.

We received 48 submitted abstracts after the call for proposal. We prioritized studies that provided empirical data and invited 19 authors to submit full papers. We then selected nine of the submitted final papers that clearly focused on K-5 CT from a range of subject areas (CS specific=1; all subjects=2; literacy=2; science=2; math=3). Five of these papers included PreK-5 student data, six included inservice teacher data, and one included preservice teacher data. This introduction highlights the themes that cut across the nine papers. In addition, we discuss the gaps that need to be investigated and provide directions for future work.

Defining Computational Thinking

The papers in this issue generally define CT based on Jeanette Wing's influential article, which stated: "Computational thinking involves solving problems, designing systems, and understanding human behavior by drawing on the concepts fundamental to computer science" (Wing, 2006, p. 33). Some papers use state or national CS standards to define CT, while others include more refined descriptions, such as the common framework PRADA (Dong et al., 2019) which includes pattern recognition, abstraction, decomposition, and algorithms. While the authors include common CT practices (such as algorithms, debugging, pattern recognition, etc.), how the authors use these CT terms manifest differently depending on their viewpoints. For example, Weintrop et al. uses PRADA for integration of CT into fourth grade mathematics, but includes iterative development and debugging practices that are associated with programming. Sheridan et al. focuses on how preservice teachers represented five CT practices (abstraction, algorithmic thinking, data, decomposition, and simulation) in their two lessons using a concept mapping tool and Scratch.

Although all papers conceptualize CT and associated practices in different ways, all provide clear examples of what the CT looked like as it was integrated into their preK-5 classroom context. The multiple ways CT plays out in the papers suggest that CT is still in its infancy and very much contextualized by researchers and teachers. As Denning (2017) argued, there are still challenges in how CT is defined and how to measure students' abilities to think computationally. The idea of CT as "algorithmic thinking" has been around since the 1950s (Caeli & Yadav, 2010; Denning, 2017; Grover, 2018) and, as evidenced by papers

in this special issue, it continues to play out in multiple ways in K-12 classrooms. As it is important to contextualize CT for particular contexts to support disciplinary learning, we want to encourage scholars in this area to clearly define CT. Perhaps even more important is the use of specific curricular examples that showcase how CT was implemented in classroom contexts. This way, readers can visualize how the authors are defining CT.

The Integration of Computational Thinking

Our K-12 curriculum is a zero-sum game, where adding a subject means something else needs to be removed. At the elementary level, the discussion of how to cover the new subject area of CT/CS has led to different implementation designs. While some suggest that CT/CS can be taught as a separate stand-alone course, such as math, reading, or science, others have suggested that elementary teachers do not have the time to teach CT or CS as its own discipline. To accommodate for these time struggles, many suggest that CT could be integrated into other core content areas (e.g., Sherwood et al., 2021). In fact, Fofang et al. (2020) indicated that the three justifications for the integration of CT include practical (e.g., lack of time), pedagogical (e.g., integration provides richer problem-solving contexts), and equity (e.g., ensuring all learners would have access). These are solid rationales for the integration of CT, and ones that other studies have shown as critical to teachers' integration of CT (Israel et al., 2015; Rich, Yadav, and Larmore, 2020).

Levels of Computational Thinking Integration

Beyond the rationale to integrate CT, however, there is a range of types of CT integration. In this issue, Coenraad et al. describes four different levels of integrating CT that were present across 22 CT-integrated science lessons created by 36 elementary teachers. These levels are

- *exist* (labeling already present CT),
- *enhance* (using CT to support science learning),
- *extend* (extend science learning by integrating CT tools and practices), and
- *exhibit* level (using programming to show science learning).

Coenraad et al. used these levels to examine how elementary teachers' science lesson plans incorporated CT and found that of the 16 lessons:

- three lessons (18.75%) integrated CT at an *exist* level,
- eight (50%) integrated CT at an *enhance* level,
- two lessons (12.5%) *extended* science learning, and
- three lessons (18.75%) integrated science and CT on an *exhibit* level.

This use of levels to describe how teachers incorporate CT is similar to that used to classify technology integration and teachers' uses of technology (e.g., Jonassen, 1996; Sandholtz et al., 1997). In this issue, Israel et al. also describe how CT integration could range "from using academic language that crosses disciplinary areas (e.g., the term decomposition across CS and math instruction) to using complex integrated computer-based activities" (p. 65). These levels or ranges of CT integration are important distinctions that need to be investigated further. Such investigations could involve questions such as "Are all levels of CT integration impactful?" and "Do these all lead to similar results?"

While the descriptive approaches that identify levels of CT integration are valuable, future research also needs to examine whether and how teachers move between integration ranges or levels. Throughout the nine papers in this issue, we found that CT integration at the lower elementary levels tended to focus on the use of the language, whereas upper elementary levels tended to connect these ideas to computer-based activities and tasks. More specifically, we want to challenge future researchers to examine how the use of CT language supports disciplinary learning, as well as whether it prepares learners for salient computationally rich applications. In other words, when we have younger students searching for patterns in language or science, or decomposing playdough art into parts, will they be able to transfer this knowledge to computer-relevant applications later? Are these appropriate tasks that we are designing and will they provide students' with strong conceptions of CT/CS later? In addition, future research should examine how CT integration supports disciplinary learning?

Furthermore, what support is needed for teachers to move from one integration approach to another and translate it into their practices? We need to examine how teachers move from unplugged CT to plugged CT, and how they implement those approaches to support their disciplinary learning goals. For example, practitioners could incorporate CT practices when teaching about the water cycle. They could do so by having students decompose the water cycle (breaking the cycle into smaller parts), discover the patterns and looping of the cycle, and write out the algorithm for the water cycle. This is a good starting place for teachers to connect CT practice in a science disciplinary context. Taking this approach one step further, however, they could have students create a model of the water cycle using computational tools (such as Scratch and SageModeler), thus providing a deeper understanding of CT concepts and practices.

Some of the papers in this issue did focus on using CT to teach core subject areas, such as mathematics, and found that teachers or students had stronger conceptions of the

other subject area content based on the integration of CT. For example, Weintrop et al. describes fourth grade students' experiences using Sphero during math lessons. The authors described how students engaged in CT practices (such as decomposition, debugging, pattern recognition, and algorithms) as well as mathematical practices (precision and proportional reasoning). Kopcha et al. also detailed a study of how elementary teachers used robotics to teach math concepts (e.g., fractions, angles, addition, perimeter and area). The authors found that, as a result of professional development on CT integration, elementary teachers' confidence in using robotics to teach math and facilitating productive math discourse was significantly higher. In Jacob et al.'s study, students showcased their storytelling and narrative skills while discussing CT concepts and dispositions in *The Most Magnificent Thing* book. The researchers found that these narrative skills were demonstrated by teaching computational thinking through literacy. Bers and colleagues also embedded CT in literacy in first and second grade classrooms using KIBO robotics. The results from this study suggest that the curriculum improved students' coding and CT skills, however students' baseline literacy skills predicted their CT skills.

Visualization of CT Integration

Regardless of how the CT was integrated, the strength of these nine papers lies in their commitment to providing detailed descriptions of the curricula. This was sometimes done using tables to outline activities and content (e.g., Jacob et al., Sheridan et al.), or through pictures of student work (e.g., Weintrop et al., Grover et al.). For example, Grover et al. showcases examples of a playdough activity whereby students were tasked with decomposing creations to identify its smaller parts. Jacob et al. provide a table showcasing the teacher activities and depict how these activities aligned with more English Language Arts standards and CSTA standards. Sheridan et al. even provides an outline of the programming module, with examples of the assignments and discussion questions. Bofferding et al. (this issue) also include all three sets of the worked examples used in their study, thus providing the pedagogical design choices within the table that illustrates the CT learning connections.

These thick descriptions help readers understand how CT is being integrated. Similar to the concerns associated with the CT definitions above, the thick descriptions also help orient the readers and allow us to view CT through the authors' lenses. This is an important concept as we work together to better understand CT, how learners conceptualize CT, and how CT can be integrated into the elementary curriculum.

Systemic Change for CT integration

In this special issue, the study by Israel et al. describes different approaches to implementing CT focused curriculum across an elementary school. The authors articulate a number of challenges for CT integration:

A. limited CT teaching expertise,
B. limited time for CT integration, lack of CT-specific assessment knowledge and tools,
C. limited pedagogical understanding for meeting students' diverse instructional needs, and
D. low teacher-buy-in for teaching CT.

These findings suggest the need for a system-wide implementation of CT to ensure that all students are introduced to CT and not only the students of teachers who decide to participate in professional learning around CT. Given that teachers face a number of challenges such as standardized testing requirements and constraints of the curriculum, it is critical for schools to have a vision for CT integration combined with support for teachers. Computer science education researchers must develop researcher practitioner partnerships that take a holistic approach to bringing CT across the curriculum in schools. Furthermore, these partnerships should extend beyond schools to include the community and family involvement. One study in this issue by Grover et al. focuses on examining how CT could be integrated into preschool math and science instruction. The authors found that involving parents/caregivers in the design of CT learning activities could be productive to engage young learners in CT practices. This provides additional evidence of the importance of engaging multiple stakeholders to ensure a successful implementation and diffusion of CT/CS at the elementary levels.

Measuring Computational Thinking

It is important to note that the papers in this issue used different approaches to measuring teachers and/or students' knowledge, skills, and attitudes towards CT. For example, Grover et al. observed preschool students in their homes, noting where they placed specific manipulatives, or whether they pointed or verbally described something. Weintrop et al. vignettes that not only described the activity the students participated in, but also provided a clear discussion of how student words and actions represented CT concepts. Jacob et al. provided a complete audio transcript of conversations between the teacher and students, demonstrating how the teacher used the popular book, *The Most Magnificent Thing*, to teach her students about debugging and iterating. Bofferding et al. used worked examples to measure students' CT understandings,

which was represented by a success score (did they achieve the objective), as well as an in-depth investigation of students' coded programs. The creative approaches used to capture and explain how CT could be represented is another addition to our conversations around this topic.

Conclusion

Overall, the nine papers included in this special issue provide a diverse overview of computational thinking at the elementary level. They showcase how to teach CT at the elementary level through a range of the following:

- Approaches: integrated and stand-alone
- Activities: unplugged manipulatives, computational toys, software applications and services
- Areas of focus: pattern recognition, algorithms, decomposition, debugging, etc.
- Instructional strategies: worked examples, hands-on practice, graphic organizers, cooperative learning;
- Grades levels: PreK-5; and
- Subject areas: math, science, language arts, CS.

The field still needs to identify developmentally appropriate practices and learning goals for elementary students. In addition, there is still a great deal of research needed to examine how to integrate CT into other subject areas, benefiting both the subject areas and CT/CS.

References

Caeli, E. N., & Yadav, A. (2020). Unplugged approaches to computational thinking: A historical perspective. *TechTrends, 64*(1). https://doi.org/10.1007/s11528-019-00410-5

Denning, P. J. (2017). Remaining trouble spots with computational thinking. *Communications of the ACM, 60*(6), 33–39. https://doi.org/10.1145/2998438

Dong, Y., Catete, V., Jocius, R., Lytle, N., Barnes, T., Albert, J., ... & Andrews, A. (2019, February). PRADA: A practical model for integrating computational thinking in K-12 education. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education* (pp. 906-912).

Fofang, J. S., Weintrop, D., Walton, M., Elby, A., & Walkoe, J. (2020). Mutually supportive mathematics and computational thinking in a fourth-grade classroom. In Gresalfi, M. and Horn, I. S. (Eds.), The Interdisciplinarity of the Learning Sciences, *14th International Conference of the Learning Sciences (ICLS) 2020*, Volume 3 (pp. 1389-1396). Nashville, Tennessee: International Society of the Learning Sciences.

Grover, S. (2018, November 5). *A tale of two CTS (and a revised timeline for computational thinking)*. ACM. Retrieved November 13, 2021, from https://cacm.acm.org/blogs/blog-cacm/232488-a-tale-of-two-cts-and-a-revised-timeline-for-computational-thinking/fulltext.

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

Hsu, T. C., Chang, S. C., & Hung, Y. T. (2018). How to learn and how to teach computational thinking: Suggestions based on a review of the literature. *Computers and Education*. https://doi.org/10.1016/j.compedu.2018.07.004

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.

Jonassen, D. H. (1996). *Computers in the classroom: Mindtools for critical thinking*. Prentice-Hall, Inc..

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

Sandholtz, J. H., Ringstaff, C., & Swyer, D.C. (1997). *Teaching with technology: Creating student-centered classrooms*. Teachers College Press.

Yadav, A., Hong, H., & Stephenson, C. (2016). Computational thinking for all: Pedagogical approaches to embedding a 21st century problem solving in K-12 classrooms. *TechTrends 60*, 565-568. DOI: 10.1007/s11528-016-0087-7.

Investigating Pre-Service Teachers' Computational Thinking Through Lesson Design Using Digital Technologies

Scott Sheridan, Bataul Alkhateeb, Chrystalla Mouza, *School of Education, University of Delaware,* **and Hui Yang,** *SRI International*

Corresponding Author: Chrystalla Mouza, *cmouza@udel.edu*

Abstract

In this chapter, we describe a pedagogical approach aimed at preparing pre-service teachers to integrate (CT) into K-8 contexts. Specifically, we present a standalone educational technology course with explicit attention on connecting CT to disciplinary content and pedagogy while introducing a range of digital tools. Data were collected from 34 pre-service teachers over the period of one semester. Specifically, a total of 68 lesson plans developed by pre-service teachers through two distinct lesson planning tasks using concept mapping and programming tools were collected and analyzed using both quantitative and qualitative techniques. The analysis utilized a coding scheme that focused on identifying specific CT practices illustrated in each lesson. It also examined differences in the CT practices exhibited in each lesson planning task. Findings indicated that the CT practice of data was most prevalent in pre-service teachers' lesson plans. Outside of data, however, there was greater variation in the CT practices represented in programming lesson plans compared to the concept mapping lesson plans. Implications for teacher educators are discussed based on the findings.

Introduction

In recent years there has been a renewed focus on the development of computational thinking (CT) among all students. CT has its origins in the work of Seymour Papert (1980) who aspired to engage all students in computer programming. Augmented by innovative low-floor high-ceiling programming languages (e.g., Scratch), CT has resurfaced as a critical 21st century skill for all students (Wing, 2006). As such, it has been incorporated in a variety of content area standards, including the Common Core State Standards in the United States (CCSS, 2010), the Next Generation Science standards (NGSS, 2013), and the National Education Technology standards (ISTE, 2016).

While CT has emerged as an area of growing significance, definitions of CT continue to vary in the literature. The most popular definition was provided by Wing (2006), indicating that CT "involves solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science" (p. 33). In the context of K-12 education, however, Barr and Stephenson (2011) defined CT as a problem-solving methodology that can be transferred and applied across subjects, which is important for K-12 as it points out the connection of CT to various subject areas. Indeed, CT has been suggested as a set of practices (e.g., algorithmic thinking, decomposition, abstraction, data, simulation) that are more than just programming and can be integrated in disciplines outside of computer science (CS) to support disciplinary learning (Barr & Stephenson, 2011; Yadav et al., 2021). Further, it can be integrated using digital tools beyond programming (e.g., concept mapping, data collection tools) beginning at the elementary level (Kotsopoulos et al., 2017; Lambrou & Reppenning, 2018).

A key challenge in the introduction of CT in K-8 education is the preparation of pre-service teachers. Yadav et al. (2017) have argued that in order for pre-service teachers to teach CT, they need to develop a deep understanding of both their content area and of CT. Indeed, research indicates that effective integration of CT necessitates that pre-service teachers build knowledge of new computing content (CK), knowledge of good pedagogical practices (PK), and knowledge of technology tools (TK) inherent in CT instruction (Mouza et al., 2017). The interactions among these knowledge domains form the core of what has been called Technological Pedagogical Content Knowledge (TPACK; Mishra and Koehler, 2006), CS-related TPACK (Vivian & Falkner, 2019) or TPACK-CT (Mouza et al., 2017). As the interest in computing is growing, teachers need support to navigate these knowledge domains (Vivian & Falkner, 2019). Yet limited

research has focused on building pre-service teacher' knowledge for CT integration across content areas.

One way to advance pre-service teacher knowledge of CT is through standalone educational technology courses required in most teacher education programs around the U.S. (Yadav et al., 2017). Yadav et al. (2011), for instance, integrated introductory CT learning modules in an educational psychology course for pre-service teachers focusing on examples of CT application in both science and humanities. Similarly Bean et al. (2015) introduced pre-service teachers to coding in three subject areas: music, language arts, and mathematics. Results indicated that pre-service teachers' knowledge and self-efficacy of CT improved. Here, we discuss one approach to integrating CT in an educational technology course. The course that served as the foundation of this work introduces computing tools and practices specific to incorporating CT with content and pedagogical knowledge in K-8 settings.

As part of their participation in the course, pre-service teachers completed lesson planning tasks and sample products (i.e., an example of what they expected students to produce) that asked them to integrate CT with disciplinary content and pedagogy using digital tools that can support the development of CT practices among K-8 students. Such tools include, among others, concept mapping software (i.e., software that allows the development of conceptual diagrams or figures illustrating suggested relationships among concepts in a domain; see also Malallah & Weese, 2020) and visual oriented programming software (i.e., Scratch). For instance, pre-service teachers could engage students in the CT practice of decomposition by asking them to visually illustrate the process of breaking down and solving a mathematics problem using a concept mapping tool. Through these lesson design tasks we sought to identify the CT practices most frequently represented by pre-service teachers and the extent in which they connected CT with content and pedagogy in the spirit of the TPACK framework.

In prior work, we have found that pre-service teachers were able to develop CT-integrated lesson plans aligned with content and pedagogy more successfully when using concept mapping tools compared to Scratch programming (Sheridan et al., 2020; Yang et al., 2018). We attributed this difference to pre-service teachers' greater familiarity with concept mapping software compared to Scratch programming (Zinth, 2016). As a result, we have redesigned a module focusing on programming in ways that allowed pre-service teachers to build greater familiarity with both the technology and its potential to support CT development across content areas. Following the redesign of the module, we investigated the following research question:

A. What CT practices are represented in pre-service teachers' lesson planning tasks when using concept mapping tools and Scratch programming?
B. To what extent do pre-service teachers connect CT to content and pedagogy when incorporating concept mapping tools and Scratch programming in lesson planning? Are there significant differences across the two lesson planning tasks?

METHODS

Participants

Participants for this study included 34 pre-service teachers (N=34) enrolled in a four-year elementary teacher education program in a Mid-Atlantic University in the United States. Graduates of the program are eligible for an elementary (K-5) teacher certification as well as a certification in special education, English as a second language (ESL) or a middle school (6-8) content area. All participants were in their sophomore, junior, or senior year and were enrolled in a required course titled, *Integrating Technology in Education* during the period of one semester. All participants were females between the age of 18-22 except for one participant who was in the age group of 27-32. Baseline data collected at the beginning of the course indicated that although pre-service teachers were not able to expressly define CT in a way it might appear in the literature, many recognized that it was a thought process used in problem solving. All but one pre-service teacher believed that CT could be integrated into the classroom though generating examples was challenging for participants.

Description of the Course: Integrating Technology in Education

Integrating Technology in Education is a required 3-credit hour course for all pre-service teachers, typically taken during sophomore or junior year. The course spans over 14 weeks and introduces pre-service teachers to technologies available for use in classroom content areas, pedagogical considerations with these technologies, and teaching and learning practices that combine the use of technologies with content and pedagogy (Mouza & Karchmer-Klein, 2015). Although the course is usually offered in a hybrid format, due to COVID-19, the course was held asynchronously online with synchronous help sessions offered via Zoom.

Given the growing attention on CS education and the need to prepare pre-service teachers to integrate CT across the curriculum, the course was previously redesigned to support the development of pre-service teachers' knowledge of CT (see Mouza et al., 2017). Towards this goal, we introduced CT practices (e.g., decomposition, abstraction) and tools appropriate for elementary instruction using relevant theoretical and empirical articles, CT resources developed by various organizations, and hands-on activities. In this work, we view CT as an interdisciplinary set of practices that can help support existing content area instruction. Our goal was to help pre-service teachers recognize, highlight,

and design CT-integrated instruction utilizing digital tools which are usable in a broad variety of content areas (e.g., concept mapping software, programming) and widely available in mainstream classrooms (Mouza et al., 2017). This approach is different from teaching CT as part of a standalone course independent of disciplinary applications (Weintrop et al., 2016). To support pre-service teachers' instructional design, we provided a series of scaffolding questions that allowed for CT-integrated lesson planning in the spirit of the TPACK framework. Specifically, pre-service teachers engaged in the design of two lesson planning tasks following the 5-step approach presented by Harris and Hofer (2009). This approach helps pre-service teachers consider the content and pedagogy of a lesson and then identify technology tools and practices, including CT practices, that could support students' learning.

Lesson Planning Task 1: Concept Mapping

The concept mapping lesson planning task asked pre-service teachers to design a lesson plan that incorporated a concept mapping in a content area. Pre-service teachers first read and reflected on a series of articles focusing on the role and importance of concept mapping to build their knowledge of technology and pedagogy. Subsequently, they were introduced to concept mapping tools (e.g., Popplet), practiced how to use such tools, and examined existing lessons integrating concept mapping for instruction. Finally, pre-service teachers engaged with the design of their own lesson plans by responding to a series of prompts following the Harris & Hofer (2009) framework which asked them to provide: (a) specific learning goals using disciplinary standards, (b) a description of how to introduce concept mapping to students, (c) a description of activities to engage students in the learning process of disciplinary content, and (d) a description of how concept mapping could be used to help students achieve the learning goals. Each lesson plan was accompanied by a copy of a sample concept map similar to what they were expecting from their students to produce (e.g., a concept map representing the plant cycle in science to illustrate abstraction or a concept map demonstrating place value in mathematics to illustrate decomposition; see also Figure 1).

Lesson Planning Task 2: Programming

The programming lesson-planning task asked pre-service teachers to design a lesson plan that incorporated Scratch programming within a curriculum content area. Prior to designing their lesson, pre-service teachers (a) engaged with a scenario-based digital simulation where they reflected and enacted conversations related to the role of CT in elementary education, (b) reviewed readings and videos, (c) reflected on the readings, and (d) practiced using Scratch programming. They also examined existing lessons on the integration of programming in elementary instruction. Table 1 presents an overview of the module.

Table 1. Description of Programming Module

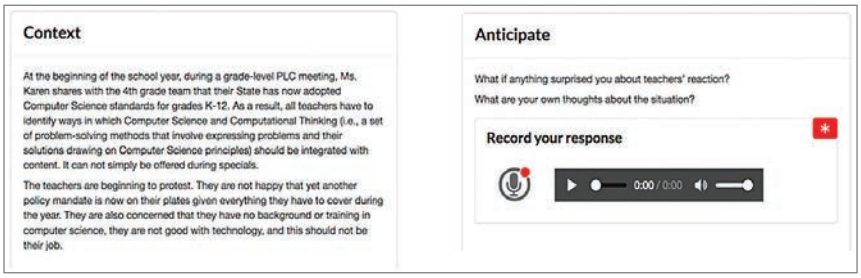
Week	Description of Course Activities
Week 1	**Scenario-Based Simulation Exercise: *CS is not my job*** Pre-Service teachers engage in a practice space: https://teacher-moments.herokuapp.com/scenarios/, which provides a space to practice ideas and consider how to respond to challenging teacher situations around CT integration. 
	Read & Watch Describe the importance of CT integration Videos explaining CT Integration (e.g., ISTE, 2012)
	Reflect: 3-2-1 (1) Identify 3 new insights or take-aways from your readings on computational thinking. (2) Discuss 2 more ways in which you can integrate computational thinking in your future classroom and the ways in which students could benefit from such integration (these examples could be very brief, e.g., use an algorithm to model the exit routine from the classroom). (3) Identify any remaining questions about computational thinking.

Table continued on next page

Table 1. Description of Programming Module *(continued)*

Week	Description of Course Activities
Week 2	**Hour of Code: Watch, Practice & Discuss (http://code.org)** Discussion Questions (1) Which programming course did you try out in Code. org? (2) What did you like/did not like about it? (3) What suggestions do you have for improving the course and what is your rationale?
	Scratch Tutorial: Let's Dance 
	Reflect (1) Identify 3 new insights or take-aways from your experience programming with Scratch. (2) How would you describe your experience with Scratch? Did anything surprise you?
Week 3	**Lesson Planning Review & Design** (1) Review CT-Integrated lessons on ScratchEd. Examples in Storytelling, Math, Science, and Music. (2) Lesson Design (Parallels concept mapping lesson design) • Select a content area of your choice: What is the learning goal? What standards does it address? • Describe the pedagogical knowledge. • Consider how you will introduce Scratch to your students. • Describe the activity types used in the lesson. • Describe the assessment strategies used in this lesson. • How does the use of programming support the learning goals and pedagogical knowledge identified in the lesson?

Data Collection

This study specifically focuses on the two lesson planning tasks to examine the integration of CT within various content areas. Each participant (N=34) completed the two lesson planning tasks described above, which resulted in a total of 68 lesson plans (N=68).

Data Analysis

All lesson plans (N=68) were analyzed qualitatively in three phases. First, a descriptive analysis was conducted by the first author to identify the content areas represented in each lesson. Second, each lesson plan was analyzed again by the first author to determine the presence or absence of specific CT practices using the coding scheme presented in Table 2. Third, each lesson was assessed using a rubric adapted from Harris et al. (2010) and used in prior work by authors (Mouza et al., 2017; Sheridan et al., 2020). The rubric provides a valid and reliable instrument that can be used to evaluate pre-service teachers' lesson plans for content, pedagogy, and technology in relation to CT. It incorporates four evaluation criteria including: (a) Fit: alignment of content, pedagogy, and digital tools to foster CT knowledge and skills; (b) *Curriculum Goals and Technologies:* (e.g., digital tools and practices that support the development of CT knowledge and skills); (c) *Instructional Strategies and Technology:* using computing tools to support teaching and learning that fosters students' CT knowledge and skills; and (d) *Technology Activities:* activities' compatibility with curriculum goals and instructional strategies.

Each criterion is scored on a numerical scale from 1 to 4, allowing each lesson plan to receive an aggregate score between 4-16. A score of 1 in any of the criteria indicates failure to meet the necessary requirements of the criterion, while a score of 4 indicates full success in meeting the requirements of the criterion. The first and second authors scored the lesson plans independently before coming together to review initial inter-rater reliability for CT practices present and scores for each criterion. Any inconsistencies were due to varying interpretations of the rubric and were discussed between the coders until an agreement was reached to ensure internal consistency. Subsequently, one quarter of the data were re-coded by both coders to determine inter-rater reliability for individual evaluation criteria, aggregate scores, and identifiable CT (Cohen, 1960). Pooled kappa scores were K=0.80 for *Fit*, K=0.88 for *Curriculum Goals*

Table 2. Coding Scheme with Definitions and Examples of CT Practices from Lesson Plans

CT Concept	Definition	Examples from Data
Abstraction	Reducing complexity to define main ideas	Students used a concept mapping tool to represent the characteristics of 2D shapes categorized by defining attributes (e.g., triangles are closed and have three sides).
Algorithms	A series of ordered steps taken to solve a problem or achieve some end.	Students utilized Scratch to draw polygons after given the coordinates for the vertices in a coordinate plane.
Data (Collection, Analysis and Representation)	The process of gathering appropriate information Making sense of data, finding patterns, and drawing conclusions Depicting and organizing data in appropriate graphs, charts, words, or images	Students used Scratch to program events from a story and extracted context clues from the story to organize events in chronological order.
Problem Decomposition	Breaking down tasks into smaller, manageable parts.	Students created their own Scratch projects breaking down three-digit numbers into tens, ones, and hundreds.
Simulation	Representation of models of a process. Simulation also involves running experiments using models.	Students developed a model to represent the various stages of Earth's minerals and rocks as influenced by the flow of energy that drives the cycle. (e.g. melting, crystallization, weathering, deformation, and sedimentation).

(Definitions of CT from CSTA & ISTE, 2011)

and Technology, K=0.80 for *Instructional Strategies and Technology*, K=0.88 for *Tech Activities*, K=0.86 for aggregate scores, and K=0.85 for identifiable CT; all scores ranged from K=0.80 to K=0.90 which indicate strong agreement (McHugh, 2012). Finally, numerical data from the rubric were analyzed quantitatively using multiple dependent paired sample t-tests to identify potential differences among the concept mapping and Scratch programming lesson plans. Scores from each individual criterion of the rubric as well as aggregate scores for each lesson planning task were used as dependent variables.

RESULTS

Subject Areas Represented in Lesson Plans

In the context of the lesson planning tasks, pre-service teachers were given autonomy for choosing the content area and disciplinary standard(s) their lesson plans addressed. An overview of the lesson plans by subject area is presented in Table 3.

Results demonstrate that pre-service teachers' lesson plans were primarily focused on the core subjects of English Language Arts (ELA) and mathematics and to a lesser extent science. Traditionally, elementary school teachers spend most of their instructional time on core academic subjects such as literacy and mathematics. Therefore, the focus on these two areas among participants is not surprising.

Table 3. Lesson Plans by Subject Area

	Concept Mapping	Scratch Programming
ELA	8	16
Math	14	9
Science	7	7
Social Studies	5	0
Music	0	2

Within the concept mapping lesson plans, five teachers created social studies lessons, and within the Scratch programming lessons, two teachers created music lessons. Nearly half of the programming lessons were created for ELA content (47%). This finding is noteworthy because many efforts to introduce CT in K-8 curricula focus primarily on mathematics and science (Lye & Koh, 2014), thereby limiting opportunities for students to consider new and diverse computing pathways through other content areas.

CT Practices Represented in Lesson Plans

Examination of the lesson plans uncovered a wide range of CT practices represented in pre-service teachers' lessons. However, there were seven instances where CT was not discernable in the programming lesson plans in contrast to the concept mapping lesson plans which all contained discernable CT practices. An overview of the CT

practices incorporated into lesson plans is presented in Table 4. As shown on Table 4, the CT practice of data was prevalent in both lesson planning tasks. Outside of data, however, there was greater variation in the CT practices represented in programming lesson plans compared to the concept mapping lesson plans.

Table 4. CT Practices Represented in Each Lesson Planning Task

	Concept Mapping Lesson	Scratch Programming Lesson*
Abstraction	6	3
Algorithmic Thinking	1	6
Data	19	7
Decomposition	6	4
Simulation	1	8

*Seven of the Scratch lesson plans contained no discernable CT concepts

Technology, CT, and Pedagogy across Lesson Plans

On average, pre-service teachers' concept mapping lesson plans received higher scores on all four criteria of the rubric (*Fit, Curriculum Goals & Technology, Instructional Strategies & Technology,* and *Technology Activities*) compared to their programming lesson plans. Average total scores for concept mapping lesson plans were also higher (10.9 vs. 9.54). Descriptive statistics based on the application of the rubric for both lesson plans are presented in Table 5.

Despite receiving higher aggregate scores and higher scores on all four criteria of the rubric, t-test results indicated that there was a statistically significant difference in concept mapping and programming lesson plan scores only on the criterion of *Fit* (MD = 0.47, t(33) = 2.14, p = 0.04). This finding indicates that participants were more likely to design lessons that illustrated alignment between content, pedagogical strategies, and use of concept mapping software in ways that fostered the development of CT knowledge and skills. There were no statistically significant differences on the criteria of *Curriculum Goals & Technology* (MD = 0.32, t(33) = 1.48, p = 0.15), *Instructional Strategies & Technology* (MD = 0.24 , t(33) = 1.14 , p = 0.26), *Technology Activities* (MD = 0.29, t(33) = 1.20, p = 0.24), and *Aggregate Scores* (MD = 1.32, t(33) = 1.53, p = 0.14).

Although there were no statistically significant differences across most of the individual criteria or aggregate scores, data indicate that a greater number of pre-service teachers developed concept mapping lesson plans which received scores in a range from 9-12. A lesson plan with a high score (13-16), for instance, illustrated the CT practice of decomposition by having students use concept mapping to demonstrate their understanding of place values in mathematics and represent three-digit values in multiple formats (see Figure 1a). On the other hand, a lesson plan with a low score (4-8) used concept mapping for strict teacher guided instruction rather than for CT-integrated student production. In contrast, Scratch programming lessons tended to receive either high scores (13-16) or low scores (4-8) with fewer tending towards the mean. A lesson plan with a high score, for instance, addressed ELA standards in which students programmed an animation simulating components needed for storytelling

Table 5. Descriptive Statistics of Rubric Scores

Lesson Plan	Fit		Instructional Strategies & Tech		Curriculum Goals & Tech		Tech Activities		Total	
	M	SD	M	SD	M	SD	M	SD	M	SD
Concept Mapping (N= 34)	2.79	0.73	2.68	0.77	2.74	0.75	2.65	0.85	10.9	2.96
Scratch (N = 34)	2.32	1.34	2.44	1.21	2.41	1.28	2.35	1.37	9.53	5.12

Table 6. Statistical Analysis (t-Test) Results

Rubric	M (Concept Mapping)	M (Scratch)	M diff	SD	t	df	p (two-tailed)	Effect size (d)
Fit	2.79	2.32	0.47	1.28	2.14	33	0.04*	0.42
Instructional Strategies & Tech	2.68	2.44	0.24	1.21	1.14	33	0.26	0.23
Curriculum Goals & Tech	2.74	2.41	0.32	1.27	1.48	33	0.15	0.30
Tech Activities	2.65	2.35	0.29	1.43	1.20	33	0.24	0.25
Total	**10.85**	**9.53**	**1.32**	**5.04**	**1.53**	**33**	**0.14**	**0.31**

*$p < .05$ and effect size <0.3 is small, 0.3-0.5 is medium, and >0.5 is large (Cohen, 1988). (N=34)

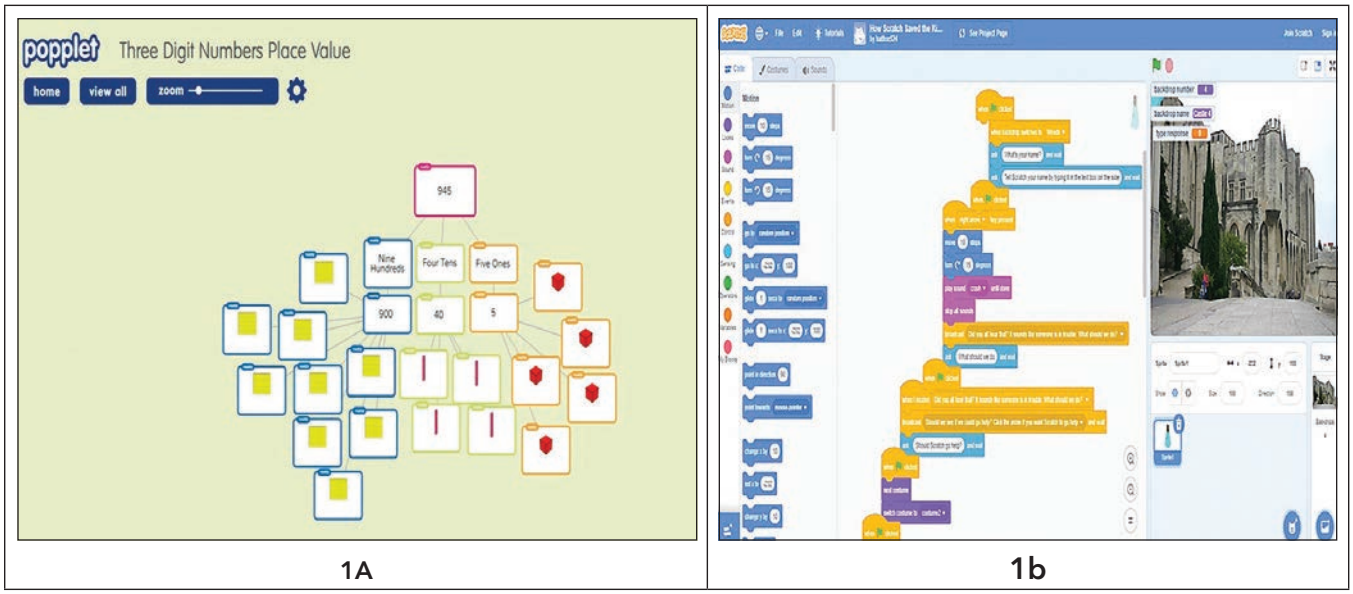


Figure 1. Decomposition with Concept Mapping (1a) and Programming Simulation in Scratch (1b) Lesson Plans

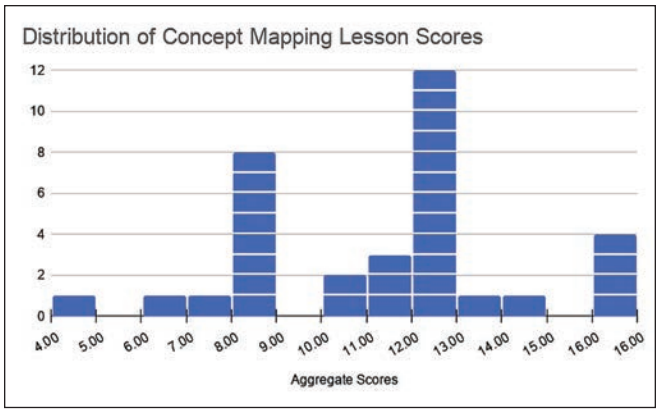


Figure 2. Score Distribution of Concept Mapping Lesson Plans Based on Rubric

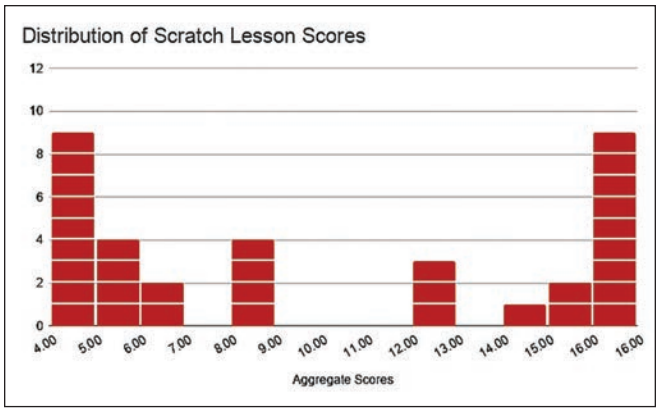


Figure 3. Score Distribution of Programming Lesson Plans Based on Rubric

(see Figure 1b). However, a lesson plan with a low score, used a premade Scratch program as a digital worksheet, disconnected from CT-integrated standards specified in the lesson plan. Figures 2 and 3 show histograms with distributions of the scores for concept mapping and programming lesson plans.

A closer examination of the Scratch programming lessons revealed that the majority of high scoring lesson plans focused on the CT practice of simulation (n=8). The lowest scoring group of lessons were those that had no discernable CT (n=7). Lessons which had CT practices other than simulation (n=19) had scores ranging from 4 to 16. Figure 4 shows a histogram with distributions of scores for Scratch lesson plans grouped by CT practice.

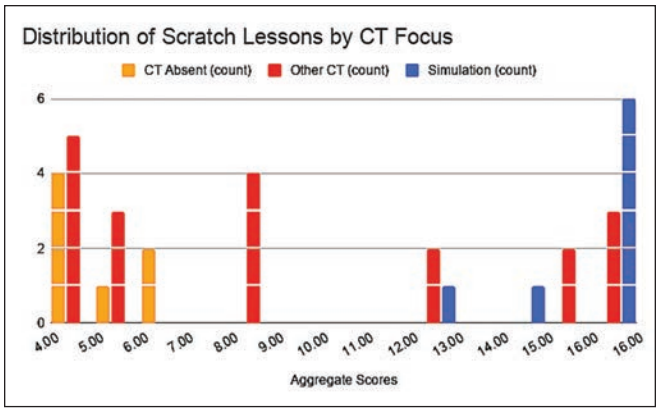


Figure 4. Distribution of Programming Lesson Plan Scores by CT Practice

Discussion

In this work we examined the ways in which pre-service teachers represented CT in two different lesson planning tasks. We also examined the ways they integrated CT with content and pedagogy when using different digital tools. Findings indicated the prevalent use of data across lesson plans, particularly lessons using concept mapping (55.9%). This finding is consistent with prior work and is largely attributed to pre-service teachers' familiarity with data through other disciplines (McGinnis et al., 2019). Concept mapping tools are primarily designed for analyzing the connections between different types of data and concepts, which are fundamental CT skills that can be used across disciplines (Psycharis, 2018). Scratch lesson plans were more likely to utilize simulations and incorporate a wider range of CT practices. These findings indicate that widely available computing tools for K-8 teachers have different affordances that could be leveraged to support distinct CT concepts. For instance, the affordances of concept mapping software make it more difficult to utilize such tools for simulations. In contrast, the dynamic nature of programming tools facilitates the design of simulations that illustrate emergent phenomena across various fields (Weintrop et al., 2016).

Examining the ways in which pre-service teachers integrated CT with content and pedagogy, findings indicated that there were no significant differences across the two types of lesson plans. In prior work, we have documented significant differences between the concept mapping and Scratch lesson plans with concept mapping lessons receiving higher scores, indicating a stronger alignment among CT, technology, content, and pedagogy (Sheridan et al., 2020). Findings from this work indicated that greater exposure to Scratch programming, as provided in the redesigned programming module, and more examples showcasing the integration of Scratch programming across subject areas may have contributed to pre-service teachers' lesson designs. Nonetheless, findings also indicated that CT was absent in some Scratch programming lessons and that lessons tended to receive either high or low scores. This indicates that pre-service teachers continued to struggle integrating programming in ways that supported both specific learning goals and pedagogy. Lessons receiving high scores typically utilized Scratch for simulation – a key affordance of programming.

Implications

This study demonstrates that developing successful CT-integrated lessons using different digital tools may be closely tied to the affordances they offer. Concept mapping tools offer affordances for connecting concepts, ideas, names, dates and other forms of data with one another, making them a good candidate for promoting use of data. Programming tools, such as Scratch, have been developed to offer different affordances such as creating animations, games, and simulations. The results from this work offer some evidence that when pre-service teachers take such affordances into account, even when they have limited technological knowledge, they can create lessons that use digital tools to create CT rich learning experiences. Nonetheless, it appears that distinct modules within educational technology coursework are not adequately building pre-service teachers' knowledge of CT in the spirit of the TPACK framework. As a result, there are three implications for educational researchers and practitioners:

A. Different tools may be able to support different types of CT practices (e.g., concept mapping software may be well suited for supporting the CT practice of data while programming may be best suited for supporting CT practices associated with algorithmic thinking or simulations). Hence, teacher educators need to help pre-service teachers leverage tool affordances in ways that optimize CT instruction and student development of associated CT practices.

B. Pre-service teachers could benefit from a greater range of CT-infused examples in relation to content and pedagogy. The ScratchEd online community (https://scratched.gse.harvard.edu/) could serve as a helpful resource, since pre-service teachers do not yet have adequate opportunities to observe the implementation of CT-infused lessons in their field placements (Mouza et al., 2017). This may help pre-service teachers envision uses of CT in content areas beyond ELA and mathematics. Further, scenario-based digital simulations, such as the one used in this work (https://teacher-moments.herokuapp.com/scenarios/), could also help pre-service teachers practice pedagogical ideas around CT. Teacher educators should incorporate such opportunities across pre-service curricula to more clearly illustrate the integration and role of CT across different content areas.

C. While designing lesson plans is beneficial for helping pre-service teachers connect CT to content and pedagogy, it is important that teacher educators also provide opportunities for implementing and reflecting on CT lessons in authentic settings to help bridge theory and practice.

References

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.

Bean, N., Weese, J., Feldhausen, R., & Bell, R.S. (2015). Starting from scratch: Developing a pre-service teacher training program in computational thinking. Paper presented at the *2015 IEEE Frontiers in Education Conference*.

Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1), 37-46.

Common Core State Standards Initiative (2010). http://www.corestandards.org/

CSTA & ISTE (2011). *CT vocabulary and progression chart*. Retrieved from: https://id.iste.org/docs/ct-documents/ct-vocabulary-and-progression-chart.pdf?sfvrsn=2

Harris, J., Grandgenett, N., & Hofer, M. (2010). Testing a TPACK-based technology integration assessment instrument. In C. D. Maddux, D. Gibson, & B. Dodge (Eds.), *Research highlights in technology and teacher education* (pp. 323–331). Society for Information Technology and Teacher Education (SITE).

Harris, J., & Hofer, M. (2009). Grounded tech integration. *Learning and Leading with Technology*, 37(2), 22-25.

ISTE. (2012, January 3rd). *Computational thinking: A digital age skill for everyone*. [Video File]. Retrieved from: https://www.youtube.com/watch?v=VFcUgSYyRPg

International Society for Technology in Education (2016). *National educational technology standards for students*. Retrieved from http://www.iste.org

Kotsopoulos, D., Floyd, L., Khan, S., Namukasa, I. K., Somanath, S., Weber, J., & Yiu, C. (2017). A pedagogical framework for computational thinking. *Digital Experiences in Mathematics Education*, 3(2), 154-171.

Lamprou, A., & Reppenning, A. (2018). Teaching how to teach computational thinking. *ITiCSE* '18, July 2–4, 2018, Larnaca, Cyprus.

Lye, S.Y. & Koh, J.S.L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in Human Behavior, 41*, 51-61.

Malallah, S., & Weese, J.L. (2020). The effects of mind maps on computational thinking. *ASEE's Virtual Conference*, June 22-26, Paper ID#28329. American Society for Engineering Education.

McGinnis, R.J., Jass Ketelhut, D., Mills,K., Hestness, E., Jeong, H., Cabrera, L. (2019). Preservice science teachers' intentions and avoidances to integrate computational thinking into their science lesson plans for young learners. Annual *International Conference of the National Association of Research in Science Teaching (NARST)*, Baltimore, Maryland, April 3, 2019.

McHugh, M. L. (2012). Interrater reliability: The kappa statistic. *Biochemia Medica*, 22(3), 276-282.

Mishra, P., & Koehler, M. (2006). Technological pedagogical content knowledge: A framework for teacher knowledge. *Teachers College Record*, 108 (6), 1017-1054. https://doi.org/10.1111/j.1467-9620.2006.00684.x

Mouza, C., & Karchmer-Klein, R. (2015). Designing effective technology preparation opportunities for preservice teachers. In C. Angeli and N. Valanides (Eds.), *Technological Pedagogical Content Knowledge: Exploring, developing, and assessing TPCK* (pp. 115-136). Springer.

Mouza, C., Yang, H., Pan, Y., Yilmaz Ozden, S., & Pollock, L. (2017). Resetting educational technology coursework for pre-service teachers: A computational thinking approach to TPACK development. *Australasian Journal of Educational Technology*, 33(3), 61-76.

Next Generation Science Standards (2013). https://www.nextgenscience.org/

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

Psycharis, S. (2018). STEAM in education: A literature review on the role of computational thinking, engineering epistemology and computational science. Computational STEAM pedagogy (CSP). *Scientific Culture*, 4(2), 51-72.

Sheridan, S., Alkhateeb, B., & Mouza, C. (2020). Examining pre-service teachers' ability to incorporate computational thinking into lesson plans: A comparison of two digital technologies. In D. Schmidt-Crawford (Ed.), *Proceedings of Society for Information Technology & Teacher Education International Conference* (pp. 95-103). Online: Association for the Advancement of Computing in Education (AACE).

Vivian, R., & Falkner, K. (2019, July). Identifying teachers' Technological Pedagogical Content Knowledge for computer science in the primary years. In *Proceedings of the 2019 ACM Conference on International Computing Education Research* (pp. 147-155).

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, 127-147.

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

Yadav, A., Stephenson, C., & Hong, H. (2017). Computational thinking for teacher education. *Communications of the ACM, 80*(4), 55–62.

Yadav A, Zhou N, Mayfield C, Hambrusch S & Korb J.T. (2011). Introducing computational thinking in education courses. In *Proceedings of the 42nd ACM technical symposium on Computer science education*, ACM, pp 465–470.

Yadav, A., Rich, K., Schwarz, C., & Larimore, R. (2021). Developing elementary teachers' competencies in integrating computational thinking ideas in classrooms: Using a toolkit as a scaffold. In C. Mouza, A. Ottenbreit-Leftwich, & A. Yadav (Eds). *Professional Development for In-Service Teachers: Research and Practices in Computing Education*. Information Age: Charlotte, NC.

Yang, H., Mouza, C., & Pan, Y. (2018). Examining pre-service teacher knowledge trajectories of computational thinking through a redesigned educational technology course. *International Conference of the Learning Sciences*, June 23-27, London, UK.

Zinth, J. (2016). Computer Science in High School Graduation Requirements. ECS Education Trends (Updated). *Education Commission of the States*.

Computational Thinking Integration in Elementary Teachers' Science Lesson Plans

Merijke Coenraad, Lautaro Cabrera, Heather Killen, Dr. Jan Plane, and Dr. Diane Jass Ketelhut,

University of Maryland - College Park

Corresponding Author: Merijke Coenraad, *mcoenraa@umd.edu*

Abstract

Due to the increasingly computational nature of professions, computational thinking (CT) is of growing importance to authentic science learning and the education of future scientists. To meet this need, CT integration is expanding within classrooms. We provided professional development (PD) for pre-service and in-service teachers focused on integrating CT into elementary science. At the PD culmination, 36 teachers wrote and enacted 22 unique CT-integrated science lessons, individually or with teaching partners. Waterman et al. (2020) suggested three levels of integrating CT within lesson plans: *exist*, labeling already present CT; *enhance*, adding CT components; and *extend*, adding activities supporting science learning with CT. Using this framework, we examined these lesson plans, their alignment to CT practices, and the level of CT integration. Our results indicated 83.3% of teachers successfully integrated CT within their lessons, focusing on Using Data, Computational Simulations, and Programming practices. Further, we found the level of integration differed by CT practice. Data practices generally led to *exist* level integration, Computational Simulation practices to *enhance* level integration, and Programming practices tended to *extend* science lessons or exhibit the science topic with Programming. Our data demonstrated teachers can write CT-integrated lesson plans, but all levels of integration are not equal opportunities for authentic scientific learning. As the field seeks to offer equitable and quality CT experiences for all students integrated within disciplinary subjects, we must understand the level of CT integration and consider how different levels of integration could affect opportunities for students.

Introduction

As society becomes increasingly computational, computational thinking (CT) instruction has taken on a growing role in schools (Bocconi et al., 2016; National Research Council, 2010). Beyond focusing on CT within computer science (CS) courses, researchers and educational standards encourage the integration of CT into disciplinary subjects (Lee et al., 2020), particularly science (NGSS Lead States, 2013). This shift towards CT integration has two goals: to provide opportunities for all students to access computing opportunities—a first step to broadening participation in computing courses—and to enhance science learning by making content more authentic to modern professional science. However, to provide CT learning opportunities that meet these two goals, it is imperative we equip teachers with the necessary knowledge and skills to integrate CT in ways that are both authentic to computing and prepare students to engage in computing as a way to learn science. It is important to prepare teachers for this task at the elementary level, where children are beginning to explore their academic and vocational identities and are impacted by positive experiences in science and computing (Tran, 2019).

Prior research has focused on preparing both pre-service and in-service teachers to integrate CT. Teacher educators have integrated CT learning into pre-service technology courses (e.g., Chang & Peterson, 2018; Mouza et al., 2017) and science methods courses (e.g., Jaipal-Jamani & Angeli, 2017; McGinnis et al., 2020). Further, researchers have encouraged focusing on both technology and disciplinary teaching (Blikstein, 2018; Yadav et al., 2017). Studies have built pre-service and in-service teacher knowledge using CT tools such as robotics (e.g., Jaipal-Jamani & Angeli, 2017), block-based programming environments (e.g., Bean et al., 2015; Bort & Brylow, 2013; Dodero et al., 2017), and simulations (e.g., Ahamed et al., 2010).

Taken together, this research has had mixed results. While both pre-service and in-service teachers showed an improved understanding of CT following professional development (PD) (Jaipal-Jamani & Angeli, 2017; Curzon et al., 2014; Yadav et al., 2014) and increased self-efficacy and attitudes about the importance of CT (Bower et al., 2017; Simmonds et al., 2019), some still had misconceptions about CT (Chang & Peterson, 2018; Lamprou & Repenning, 2018) and had difficulty writing lesson plans integrating

CT concepts and tools into disciplinary contexts (Bort & Brylow, 2013; Mouza et al., 2017). Therefore, it is important to investigate how best to support both pre-service and in-service teachers in enacting their knowledge of CT when designing integrated lessons and implementing them in their classrooms.

Our work examined how pre-service and in-service elementary teachers (herein referred to as "teachers") learned to integrate CT into elementary science teaching through PD. We focused on supporting teachers to write and enact CT-integrated lesson plans. To support our instruction, we developed the Framework for Teachers' Integration of Computational Thinking into Elementary Science specifically focused on supporting elementary teachers within a CT for science perspective (Ketelhut et al., 2019). Based on Weintrop et al. (2016) and informed by our prior work with teachers, the framework was designed with three main considerations: eliminating computer science jargon teachers found inaccessible, selecting practices elementary-aged children could engage in, and differentiating CT from scientific inquiry—a distinction we found was blurry for teachers. The framework divides CT into four sets of practices: Using Data, Programming, Computational Simulations, and Systems Thinking from a CT Perspective (Figure 1; for detailed definitions of each practice, see Cabrera et al., 2021). We introduced the framework to teachers early in their PD and used it throughout the program as a definition of CT and a set of concrete practices students should engage in during science learning. Within this paper, we also used this framework when analyzing teachers' enactments of CT in their classrooms.

Within the PD, we focused on supporting pre-service and in-service teachers to write and enact CT-integrated lesson plans within their elementary science classes. In this paper, we will present the lesson plans teachers developed and examine the CT practices integrated and the level of CT integration achieved according to a framework designed by Waterman et al. (2020). We aim to determine the level of CT integration in the lessons and identify patterns of integration to answer the research question: *To what extent do pre-service and in-service teachers integrate computational thinking into elementary science lesson plans?*

METHODS

We designed and implemented the *Science Teaching Computational Thinking Inquiry Group* (STIG^{CT}) to collaborate with teachers around integrating CT in elementary science lessons within a community of practice (Coenraad et al., 2021; PD guide and activities at https://education.umd.edu/stigct). STIG^{CT} was a semester-long PD for both pre-service and in-service teachers developed through Design-Based Research (Brown, 1992; Barab, 2006). Teachers and researcher facilitators met for four 165-minute sessions (February - May 2019; 11 hours total). Prior to participating in STIG^{CT}, all teachers were introduced to CT through their pre-service science methods course or a workshop for in-service teachers. The course and the workshop were designed to cover the same information to ensure that both groups received equal grounding in CT. This included an introduction to CT and the Next Generation Science Standards, presentation of our CT framework, and completing plugged and unplugged CT activities.

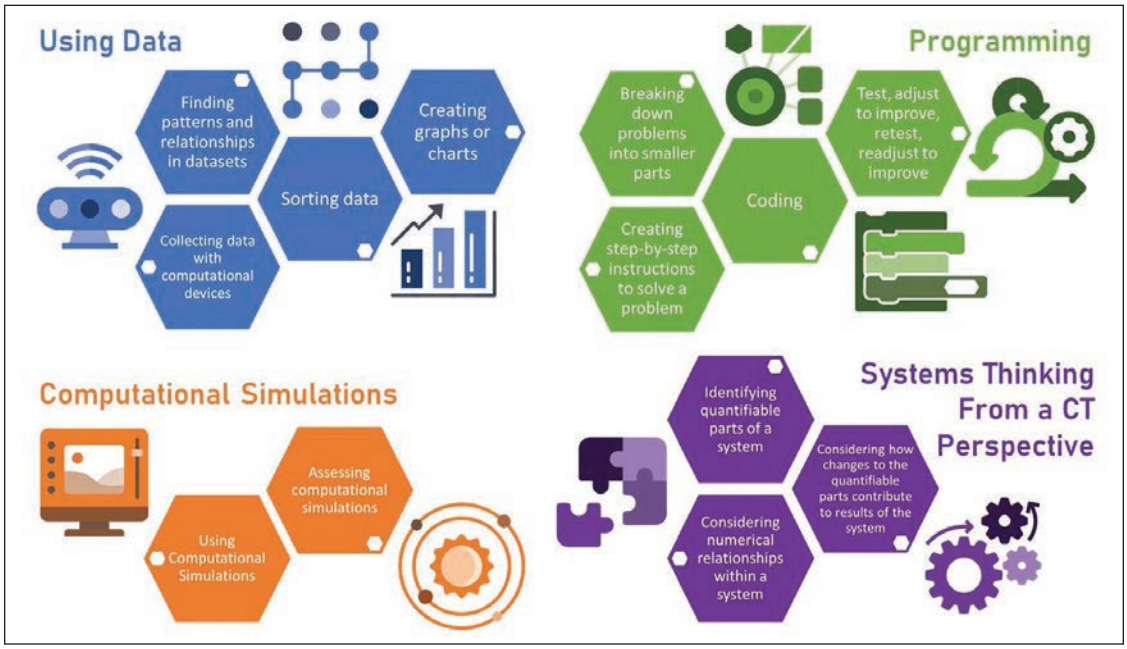


Figure 1. Framework for Teachers' Integration of Computational Thinking into Elementary Science

Each STIG^{CT} session focused on one of the four CT practices and included three sections: presentation of the CT practice, CT-integrated science activities from a student lens, and development of a lesson seed (the beginning of a lesson plan) with grade-similar peers and a facilitator. Each teacher selected one lesson seed and developed it into a full lesson plan they taught to their class. In the final session, teachers shared and reflected upon their lesson plan and teaching experience.

In total, 36 teachers participated in STIG^{CT} and submitted a lesson plan (20 pre-service, 16 in-service). They taught in elementary schools in the Mid-Atlantic region of the United States. Because some participating pre-service and in-service teachers worked together as a mentor/mentee pair or on a grade-level team at the same school, the teachers developed 22 unique lesson plans. In this paper, we analyze these lesson plans using Waterman et al.'s framework (2020) to determine the level of CT integration.

To categorize the level of CT integration within lessons, we modified Waterman et al's. (2020) three-part framework: *exist*, *enhance*, *extend* (Figure 2). In their framework, lessons are considered to be at the exist level if the "CT concepts, skills, and practices already exist in the lesson and can simply be called out or elaborated upon" (Waterman et al., 2020, p. 54). As seen in Figure 2, this is an instance of CT within a science lesson, but additional science learning is not supported by the CT. This level identifies ways CT is already in the curriculum and can act as a base for deeper integration. In their second level, *enhance*, CT is integrated based on the "creation of additional tasks or lessons to *enhance* the disciplinary concept and provide clear connection to computing concepts that are present" (Waterman et al., 2020, p. 55). In these lessons, students go beyond what is already in the curriculum, utilizing CT skills in service of their disciplinary learning. In Figure 2, this is represented by multiple instances of CT expanding the science lesson. In their final level, *extend*, teachers add CT activities, typically programming (Waterman et al., 2020). In our interpretation, we looked for teachers using CT to promote science learning through computational activities, *extending* students' learning of a disciplinary concept. As shown in Figure 2, the science lesson is expanded by the CT focus within the lesson. In addition to the levels of integration presented by Waterman et al. (2020), we included a fourth category, *exhibit*, which was identified inductively during our coding process. Lessons in this category used a CT activity, typically programming, to exhibit science knowledge students gained through other means. For example, creating a Scratch animation about an animal in its habitat based on book or online research. This can be seen in Figure 2 where CT and science are both present, but do not overlap.

During our analysis we first identified the CT practices integrated into each lesson plan. Two researchers read 20% of the data and reached 85.7% interrater reliability. The two researchers then discussed all disagreements to reach 100% agreement. One researcher then coded the remaining lesson plans. In a second round of coding, we used Waterman et al.'s (2020) framework to label each lesson plan as *exist*, *enhance*, or *extend*. Two researchers coded a subset of the lessons, discussed discrepancies in the coding, and completed the coding after reaching agreement. Following this coding, a portion of the lesson plans were identified as not aligning to any of Waterman et al.'s categories. We therefore added a fourth category for these lessons (exhibit) and re-coded the lesson plans. The initial agreement between the researchers was 90.9% (20 of 22 lessons), which was elevated to 100% after discussing discrepancies.

RESULTS

Overall, teachers were able to effectively incorporate CT into an elementary science lesson plan following their participation in STIG^{CT}. Thirty of the 36 teachers (83.33%) submitted a lesson plan containing at least one CT practice (16 of the 22 unique lesson plans; 73.73%). Within the 22 unique lesson plans, researchers identified the use of three of the four CT practices (Figure 3): Using Data (9 lessons; 40.91%), Computational Simulations (8 lessons; 36.36%), and Programming (7 lessons; 31.82%). No lesson plans contained Systems Thinking from a CT Perspective. Six lesson plans (27.27%) included no CT, despite teachers self-identifying practices in the lesson. Some lesson plans contained practices from multiple categories of CT practices (Figure 3). This overlap was most common

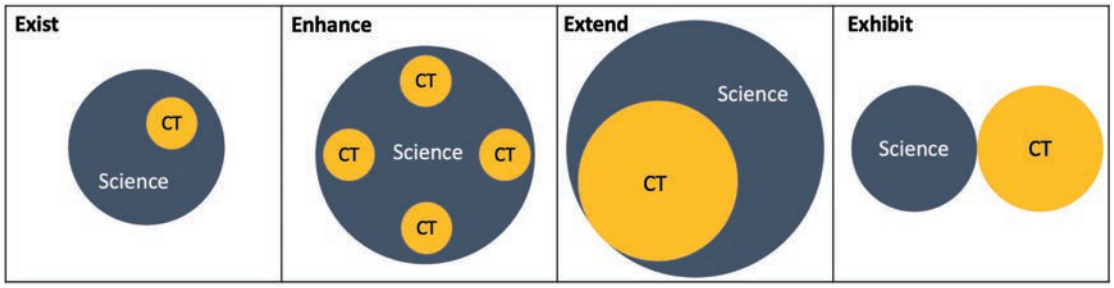


Figure 2. Levels of CT Integration

between Using Data and Computational Simulations. In these lessons, students typically collected data using a computational simulation and analyzed that data for patterns and trends to make conclusions (see Table 1 Example B below).

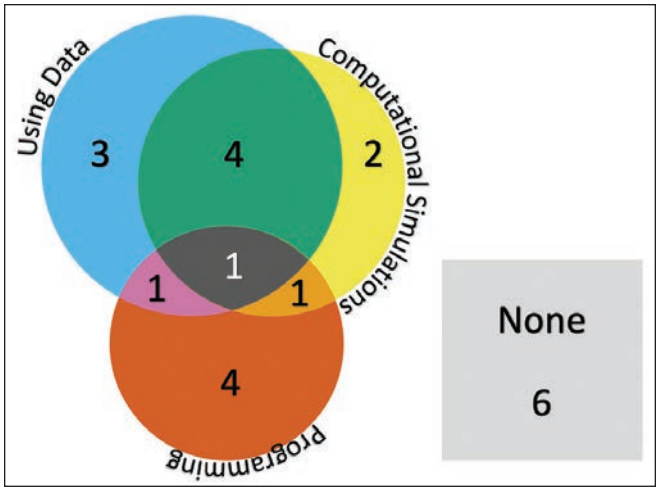


Figure 3. CT practices identified within teacher lesson plans

While teachers who participated in STIG^{CT} were able to integrate CT practices within their lesson plans generally, prior research has demonstrated not all CT integration within lesson plans provides students with equal opportunities to deeply engage with CT (Bort & Brylow, 2013; Mouza et al., 2017). Of the 16 unique teacher lesson plans containing at least one CT practice, three lessons (18.75%) integrated CT at an *exist* level, identifying CT already present within a typical science lesson plan. The greatest number of lessons, eight (50%), integrated CT at an *enhance* level, using CT to support science learning by adding CT experiences with computing tools or practices (Figure 4). Two lessons (12.5%) *extended* science learning through the integration of CT tools and practices. Finally, three lessons (18.75%) integrated science and CT on an *exhibit* level, using a relevant science concept as the topic exhibited through the CT learning experience, but doing so in a way that does not explicitly increase science learning and could be replaced with a different disciplinary topic without changing the activity.

We also found patterns between the CT practices within lessons and the level of CT integration (Figure 4). Lessons identified to include *only* Using Data practices were all at an *exist* level of integration (3 of 3 lessons; Table 1 Example A). Further, those that included Computational Simulation practices integrated CT at the *extend* level (8 of 8 lessons; Table 1 Example B). Lessons that integrated Programming practices tended to reach the *extend* (2 of 2 lessons; Table 1 Example C) or *exhibit* (3 of 3 lessons; Table 1 Example D) level of integration. These trends point to a relationship between the CT practices enacted within a lesson and the level of integration the lesson reached.

Discussion

Overall, teachers successfully integrated CT into their elementary science lesson plans across most CT practices. But integrations varied in level of integration and coverage of CT practices. Our results show that no teachers integrated Systems Thinking from a CT Perspective into their lesson plans. This raises questions about whether systems thinking around quantitative relationships is developmentally appropriate for students at the elementary level and whether further support is needed for teachers to feel confident engaging their students in discussions of systems thinking. It is common for elementary classes to examine systems such as the food web or the water

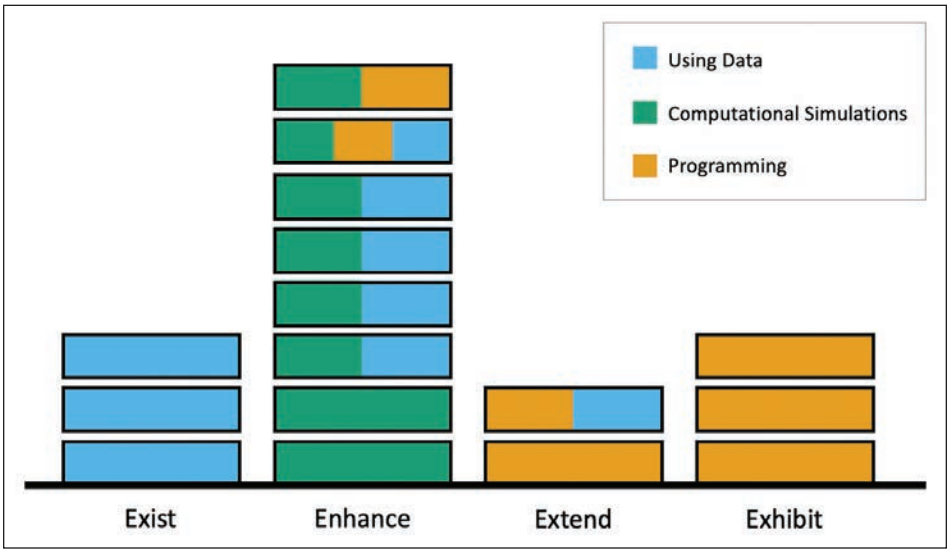


Figure 4. CT Integration levels in elementary science lesson plans

Table 1. Example lesson plans at each level of integration

Example	Level of Integration	Grade and Topic	CT Practices Integrated	Lesson Activities
A	Exist	1st grade Growing Lima Beans	**Using Data:** • Creating graphs or charts • Finding patterns and relationships in datasets	Students dissect a lima bean while making qualitative observations. Then, students plant two lima beans and place one in a lit environment (i.e., the window sill) and one in a dark environment (i.e., a dark room). Every two days, students measure and graph the height of their plants. As a class, students draw conclusions about light and plant growth based on their data.
B	Enhance	4th grade Energy Transfer	**Using Data:** • Collecting data with computational devices **Computational Simulations:** • Using computational simulations	The class reviews the vocabulary term *collision* and discusses the types of energy involved in collisions. Then, students use the PhET online simulation *Collision Lab* to collect data about two objects colliding. Students use the simulation to manipulate variables like the mass of the objects and collect data about each collision for analysis.
C	Extend	5th grade Water Pollution	**Programming:** • Coding **Using Data:** • Collecting data with computational devices	Students code a micro:bit to detect light levels. Then, using their own water samples, students measure how much light passes through water from a flashlight. Students record their data and analyze it using guided questions to make conclusions about pollution levels within the body of water from which they took their sample.
D	Exhibit	3rd grade Weather Animation	**All Programming Practices**	Students are introduced to extreme weather with an introductory video and sharing their own experiences. Then, students explore an extreme weather event by conducting guided research. To present their research, students create a Scratch animation "movie" telling about their weather event.

cycle, but these investigations rarely reach the point of interrogating the quantitative relationships within the system or representing the system using a computational tool. Future research could investigate whether elementary students can engage in systems thinking practices and develop strategies for teachers to integrate Systems Thinking from a CT Perspective.

The relationships between integration levels and CT practices we found provide insight into current gaps in knowledge and possibilities for future research around supporting teachers to write lessons with deeper levels of integration. We found that teachers who included only *Using Data* practices integrated CT and science at the *exist* level (3 of 3 lessons). This finding can partially be explained by the likeness between CT data practices and scientific inquiry, where students collect and analyze data. Further research could explore how PD can support teachers in leveraging computational aspects of data collection and analysis to move beyond *exist* level integration and into the *enhance* and *extend* levels. This effort is particularly important given that simply naming existing activities aligned with CT practices is unlikely to lead to instructional change and, therefore, new computational learning opportunities.

All lessons at the *enhance* level of integration utilized Computational Simulations (8 of 8 lessons). As the most popular tool used by teachers within their lesson plans, simulations appear to be a comfortable computational tool for integration. Yet, while teachers seemed comfortable integrating pre-made online simulations, they did not create their own simulations. Our findings suggest that integrating simulations can be an important starting point for teachers to integrate CT that can enhance science learning and inquiry. These integrations are particularly productive when studying scientific phenomena that are temporally too far away or spatially too small or large to see. However, future research could examine how teachers who feel comfortable integrating pre-made simulations could be supported to integrate CT more deeply by assessing and creating simulations with their students. While there is some important work on how students can engage with these practices (Basu et al., 2016; diSessa, 2000; Wilensky & Rand, 2015), the support that teachers need to venture into the *extend* level with simulations is less clear.

Regarding Programming practices, our findings show teachers need support to differentiate between programming activities that *extend* science learning (2 lessons) and those that only integrate science on an *exhibit* level (3 lessons). Although not included in the original Waterman et al. (2020) framework, we found exhibit to be a unique level of integration, representing the integration of science as a thematic topic in CT activities without learning-supportive integration. While the creation of a Scratch animation about a science topic is a valuable exercise to learn programming skills, the activity does not support further science learning—it can only serve as an *assessment* of content understanding. The emergence of the *exhibit* category raises questions about the integration of Scratch within CT lessons. As a tool utilized during our PD, teachers had some familiarity with Scratch. Because it is a programming environment, demonstrating a clear connection to coding and CS, the addition of Scratch was a clear-cut way for teachers to ensure they were integrating CT practices. Yet, the propensity to do so at a topic level rather than using more advanced computing such as conditionals or programming a simulation points to teachers potentially lacking confidence or knowledge with either programming tools, science, or both. To support teachers in making the differentiation between *extend* and *exhibit*, PD opportunities could include demonstrating examples of each and the differences in CT and science integrated learning they promote and providing further examples of programming that supports scientific learning.

The varied levels of CT integration highlight a need to examine the implications of different integration levels for equal and inclusive CT opportunities for students. As our findings demonstrate, even with PD focused on integrating CT in science, teachers have varied success writing CT-integrated science lesson plans. This has implications for the students in their classrooms, particularly because focusing on providing greater access to CT within classrooms is not enough to ensure equitable CT experiences for all students (Coenraad et al., 2020). If some teachers integrate at the *exist* level and others at the *enhance* or *extend* level, students are getting different levels of integration and thus different levels of preparation for the use of computing in jobs both within and outside of CS. Future research should consider the connections between school environment and teachers' level of CT integration to further understand the inequalities that could be perpetuated by different levels of CT integration. Integrating CT into science provides opportunities for *more* students to experience CT than if students only received instruction in elective or after school programs. However, as the field works toward providing equitable computing opportunities for all students, paying attention to the level of integration and the practices teachers are integrating will ensure quality opportunities for students.

Implications

Due to the nature of our study and the data we examine, implications of our work are particularly relevant to teacher educators as practitioners responsible for supporting teachers as they learn to integrate CT into disciplinary lessons. When planning and implementing PD, teacher educators should:

- Focus on supporting teachers to integrate CT in service of science learning rather than only building CT or CS skills.
- Provide explicit discussions of the levels of CT integration and moving beyond finding CT within the existing science curriculum.
- Present examples of Programming activities integrating CT at the *exhibit* and *extend* levels to demonstrate the differences in CT and science integrated learning they promote.
- Build teacher efficacy and confidence with programming environments to build their own simulations and lead students to use programming to increase understanding of science phenomena.
- Examine the barriers to integration teachers are facing and support them integrating CT at the *enhance* and *extend* levels despite the barriers they might face to provide more equitable learning experiences across CT practices for all students.

References

Barab, S. (2006). Design-Based Research: A methodological toolkit for the learning scientist. In R. K. Sawyer (Ed.), *The Cambridge Handbook of the Learning Sciences* (Issue 10, pp. 153–169). Cambridge University Press.

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

Bean, N., Weese, J., Feldhausen, R., & Bell, R. S. (2015). Starting from scratch: Developing a pre-service teacher training program in computational thinking. *Proceedings of 2015 IEEE Frontiers in Education Conference (FIE)*. https://doi.org/10.1109/FIE.2015.7344237

Blikstein, P. (2018). *Pre-college computer science education: A survey of the field*. https://goo.gl/gmS1Vm.

Bocconi, S., Chioccariello, A., Dettori, G., Ferrari, A., Engelhardt, K., Kampylis, P., & Punie, Y. (2016). Developing computational thinking in compulsory education: Implications for policy and practice. In *JRC Science for Policy Report*. https://doi.org/10.2791/792158

Bort, H., & Brylow, D. (2013). CS4Impact: Measuring computational thinking concepts present inCS4HS participant lesson plans. *Proceedings of the 44th ACM Technical Symposium on Computer Science Education (SIGCSE '13)*, 427–432.

Bower, M., Wood, L., Lai, J., Howe, C., Lister, R., Mason, R., Highfield, K., & Veal, J. (2017). Improving the computational thinking pedagogical capabilities of school teachers. *Australian Journal of Teacher Education, 42*(3), 53–72.

Brown, A. L. (1992). Design experiments: Theoretical and methodological challenges in creating complex interventions in classroom settings. *Journal of the Learning Sciences, 2*(2), 141–178. https://doi.org/10.1207/s15327809jls0202_2

Cabrera, L., Ketelhut, D. J., Mills, K., Coenraad, M., Killen, H., & Plane, J. (2021). *Designing a Framework for Teachers' Integration of Computational Thinking into Elementary Science*. [Manuscript submitted for publication].

Chang, Y., & Peterson, L. (2018). Pre-service Teachers ' Perceptions of Computational Thinking. *Journal of Technology and Teacher Education, 26*(3), 353–374.

Coenraad, M., Cabrera, L., Byrne, V., Killen, H., Ketelhut, D. J., Mills, K. M., & Plane, J. (2021). *STIGCT: The Design of a Science Teaching Computational Thinking Inquiry Group to Promote CT Integration in Elementary Science*. [Manuscript submitted for publication].

Coenraad, M., Mills, K., Byrne, V. L., & Ketelhut, D. J. (2020). Supporting teachers to integrate computational thinking equitably. *Proceedings of 2020 Research on Equity and Sustained Participation in Engineering, Computing, and Technology, RESPECT 2020*. https://doi.org/https://doi.org/10.1109/RESPECT49803.2020.9272488

Curzon, P., McOwan, P. W., Plant, N., & Meagher, L. R. (2014). Introducing teachers to computational thinking using unlugged storytelling. *Proceedings of the 9th Workshop in Primary and Secondary Computing Education (WIPSCE 2014),* 82–92. https://doi.org/https://doi.org/10.1145/2670757.2670767

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

Dodero, J. M., Mota, J. M., & Ruiz-Rube, I. (2017). Bringing computatonal thinking to teachers' training: A workshop review. P*roceedings of the 5th International Conference on Technological Ecosystems for Enhancing Multiculturality*, 1–6. https://doi.org/10.1145/3144826.3145352

Jaipal-Jamani, K., & Angeli, C. (2017). Effect of robotics on elementary preservice teachers' self-efficacy, science learning, and computational thinking. *Journal of Science Education and Technology, 26*, 175–192. https://doi.org/10.1007/s10956-016-9663-z

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

Lamprou, A., & Repenning, A. (2018). Teaching how to teach computational thinking. *Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education - ITiCSE 2018*, 69–74. https://doi.org/10.1145/3197091.3197120

Lee, I., Grover, S., Martin, F., Pillai, S., & Malyn-Smith, J. (2020). Computational thinking from a disciplinary perspective: Integrating computational thinking in K-12 science, technology, engineering, and mathematics education. *Journal of Science Education and Technology, 29*(1), 1–8. https://doi.org/10.1007/s10956-019-09803-w

McGinnis, J. R., Hestness, E., Mills, K., Ketelhut, D., Cabrera, L., & Jeong, H. (2020). Preservice science teachers' beliefs about computational thinking following a curricular module within an elementary science methods course. *Contemporary Issues in Technology and Teacher Education, 20*(1), 85-107.

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

National Research Council. (2010). *Report of a workshop on the scope and nature of computational thinking*. http://www.nap.edu/catalog/12840

NGSS Lead States. (2013). *Next Generation Science Standards: For States, By States*. http://www.nextgenscience.org

Simmonds, J., Gutierrez, F. J., Casanova, C., Sotomayor, C., & Hitschfeld, N. (2019). A teacher workshop for introducing computational thinking in rural and vulnerable environments. *SIGCSE '19 - Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, 1143–1149. https://doi.org/10.1145/3287324.3287456

Tran, Y. (2019). Computational Thinking Equity in Elementary Classrooms: What Third-Grade Students Know and Can Do. *Journal of Educational Computing Research*, 073563311774391. https://doi.org/10.1177/0735633117743918

Waterman, K. P., Goldsmith, L., & Pasquale, M. (2020). Integrating computational thinking into elementary science curriculum: an examination of activities that support students' computational thinking in the service of disciplinary learning. *Journal of Science Education and Technology, 29*(1), 53–64. https://doi.org/10.1007/s10956-019-09801-y

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

Wilensky, U., & Rand, W. (2015). *An Introduction to Agent-Based Modeling: Modeling Natural, Social, and Engineered Complex Systems with NetLogo*. Cambridge, Massachusetts; London, England: The MIT Press. doi:10.2307/j.ctt17kk851

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

Yadav, A., Stephenson, C., & Hong, H. (2017). Computational thinking for teacher education. *Communications of the ACM, 60*(4), 55–62. https://doi.org/10.1145/2994591

The Effect of Play and Worked Examples on First and Third Graders' Creating and Debugging of Programming Algorithms

Laura Bofferding, Sezai Kocabas, Mahtob Aqazade, Ana-Maria Haiduc, and Lizhen Chen,
Purdue University

Corresponding Author: Laura Bofferding, *LBofferd@purdue.edu*

Abstract

Although learning to program can improve students' computational thinking skills—specifically, creating and debugging algorithms—we need to determine instructional strategies that foster such skills in young students. We investigated the role of analyzing worked examples that focused on creating and debugging programming algorithms with 28 first and 27 third graders. Students played a tangible, block-based programming game, *Coding Awbie*, across six, 20-minute sessions and were randomly assigned to also analyze programming worked examples during sessions one to three (immediate group) or sessions four to six (delayed group). To measure changes in creating and debugging algorithms before and after their worked example intervention, students completed a pretest, midtest (after session three), and a posttest. By midtest, students who were in the immediate group wrote significantly more accurate programs, although both groups had similar accuracy when given the chance to debug their programs. By posttest, both groups made significant gains in accuracy of their programs. Overall, analyzing worked examples proved to be a powerful support for students' programming skills in creating and debugging algorithms. However, students' common bugs indicate that additional worked examples should focus on double-counting errors.

Introduction

Computational thinking (CT) is an important skill, even at the elementary level (Wing, 2006), and has received increased attention due to promotion of teaching computer science in schools and the incorporation of standards related to CT in science (NGSS Lead States, 2013) and mathematics (Pérez, 2018). In particular, creating and debugging algorithms are CT skills (K-12 Computer Science Framework, 2016) that translate across all areas of life (Yadav et al., 2016). Because the focus of most early studies with young students was on whether they *could* learn to program (e.g., Wyeth, 2008; Wyeth & Purchase, 2002), there has been less focus on investigating the methods that best promote young students' CT skills, including their debugging practices. Such investigations are needed to provide teachers with instructional activities and methods that could best support students' debugging practices. When designing our study, we focused our attention on two potentially impactful methods to support young students' programming and debugging: play and worked examples.

Learning and Debugging Through Play

According to constructivist views of learning, play provides a rich context in which children construct knowledge by exploring concepts and building on prior experiences (Ginsburg, 2006; Parks & Graue, 2015). In a programming context, play could involve children experimenting with sequencing blocks to see how robots' movements change (Highfield, 2015). One study found that young children using KIBO tangible blocks to program a robot to do dance moves improved their sequencing ability; after playing, they scored high on placing KIBO blocks in the correct order to match a story of a robot's movements (Sullivan & Bers, 2018). Children may also naturally debug during their play. Preschoolers playing with KIBO could identify and fix a program with a missing block or incorrect sequence (McLemore & Wehry, 2016). Further, some students successfully used the step-by-step debugging feature of the Robo-Blocks system to help them find their code's bugs (Sipitakiat & Nusen, 2012). This process of comparing the input with the output step-by-step, *tracing the code*, is a common debugging

strategy (Murphy et al., 2008). Given the potential benefits of play, we included opportunities for students to play with programming in our study.

However, students may struggle to successfully debug through play alone (McLemore & Wehry, 2016; Sipitakiat & Nusen, 2012). *Tinkering* or randomly changing code, which could arise during play, is usually unsuccessful as a debugging strategy (Murphy et al., 2008). In one study, third graders playing with tangible programming blocks were not always successful in their debugging because they were not sure how to evaluate whether the programs produced a desired output (Wyeth, 2008). Moreover, students navigating a robot with Robo-Blocks often thought the last block was the bug, especially when an early bug in the code was the cause of later problems (Sipitakiat & Nusen, 2012).

Learning and Debugging with Worked Examples

Another potential approach to helping students make sense of programming code is using worked examples. Worked examples present students with a set of steps used to solve a problem and can highlight common programming bugs (Griffin, 2016; Joentausta & Hellas, 2018; see also Atkinson et al., 2000; see Table 2 for examples). Given worked examples' step-by-step nature, we hypothesized that they could help students use the *tracing the code* debugging strategy (Murphy et al., 2008). Further, studying incorrect worked examples has helped older students program with fewer bugs and debug a series of steps in an algorithm (Griffin, 2016). Therefore, we chose to include opportunities for students to analyze both correct and incorrect worked examples in our study.

Although worked examples are particularly helpful for novices, if novices do not recognize what information in worked examples is important to the problem-solving task, analyzing worked examples could be misleading (Margulieux et al., 2016) or unhelpful (Ichinco et al., 2017). For example, Ichinco et al. (2017) hypothesized that students in their study (ages 10-15) either did not know what parts and commands to focus on in the examples or did not know how to map them to the programming task. Using subgoal labels (i.e., labels to help explain different steps in the code) accompanied by explanation prompts can help students overcome these issues (Joentausta & Hellas, 2018; Margulieux et al., 2016; Yan & Lavigne, 2014). In a study with third graders using LightBot, students who studied correct worked examples with subgoal labels when they encountered programming difficulties completed more levels of programming challenges than students using worked examples without subgoal labels (Joentausta & Hellas, 2018). We included both subgoal labels and explanation prompts in our study, hypothesizing they could help students use an *understanding the code* debugging strategy, which involves reasoning about what the code is doing (Murphy et al., 2008).

Current Study

Although worked examples can support older students' programming, there is scant evidence (an exception is Joentausta & Hellas, 2018) on whether a similar approach could be fruitful for early elementary students as compared to playing around with a programming game. Therefore, we set up our study design to compare the effects of playing a programming game with versus without the support of analyzing worked examples. Given that some novices have difficulty using worked examples (Ichinco et al., 2017), we also investigated whether analyzing the worked examples would be more productive for younger students' programming and debugging as they were beginning to play with the programming game versus after they had played with the game for three sessions.

When choosing the programming environment to use with the first and third graders in our study, we looked toward prior studies. Algorithmic thinking and debugging activities for younger children largely involve students learning to use tangible, block-based programming tools (e.g., KIBO blocks; Sullivan & Bers, 2018). These tangible programming blocks typically control some type of robot (tangible output) and have been used successfully with students as young as preschool (Elkin et al., 2016). Closer to traditional programming, studies with older students (ages 9-12) involve digital output, where tangible blocks control a character on a screen (e.g., AlgoBlocks in Suzuki & Kato, 1995; see also Wang et al., 2013), and non-tangible block-based programming environments with digital output (e.g., ScratchJr). Given the success of tangible materials, we chose to use the programming game, *Coding Awbie*™, which uses tangible blocks from the *Osmo* system and involves a digital output that helped us provide a consistent programming environment.

In summary, we explored how first- and third-grade students' programs and debugging changed on a tangible, block-based programming task, depending on if they analyzed worked examples during their first three out of six play sessions (*immediate group*) or during their last three play sessions (*delayed group*). Specifically, we explored the following research questions: (1) What is the effect of analyzing worked examples versus playing on first and third graders' programming accuracy? (2) What are students' common programming bugs? How do students debug them? This study takes an important step toward informing teachers about ways to effectively support young learners' creating and debugging of algorithms by focusing on a potentially impactful instructional tool: worked examples.

METHODS

Participants and Design

This paper uses data from the second year of a two-year study on exploring factors for effective commenting and debugging using the tangible programming game, *Coding Awbie*™. We worked with 28 first- and 27 third-grade students from one public elementary school in the midwestern US with approximately 45% qualifying for free and reduced-price lunch and about 11% classified as English-Language-Learners. Students first completed a pretest, after which they were assigned to one of two groups: students who analyzed worked examples during their first three play sessions (*immediate group*) or students who analyzed worked examples during their last three play sessions (*delayed group*). By having two orders in which students analyzed worked examples, we were able to determine the effect of using worked examples (WE) as an instructional tool for novices before they had a chance to play versus after they had played with a programming game (see Table 1 for the study design). For example, students in the delayed group spent the first three sessions in grade-level pairs only playing the game. Almost all activities took place with the researchers at tables outside of students' classrooms; the only exception was the programming presentation, which took place in a media room with multiple classes of students (both participants and non-participants) attending with their teachers at a time. Although several of the third graders had tried *Coding Awbie*™ prior to the study, we presented all materials assuming no knowledge of the game or programming.

Analyzing Worked Examples

Students analyzed a set of two or three worked examples (see Table 2) during sessions 1-3 for the immediate group and sessions 4-6 for the delayed group (see Table 1). The worked examples were modeled after those developed by Julie Booth and her colleagues in TheAlgebraByExample Team (n.d.) and ways of presenting worked examples in programming (Griffin, 2016; Skudder & Luxton-Reilly, 2014). We designed the worked examples to address common challenges that students encountered during the first year of our study. Based on literature showing the advantages of incorrect worked examples, we included both a correct worked example and an incorrect worked example in each set. Further supporting good debugging practice, each example included explanation prompts that asked student pairs to identify why certain code was used, what it did, or what the bug in the program was. Finally, students had to apply ideas from the examples to a new task (i.e., Your

Table 1. Study Design

Activity	Format	Immediate Group	Time (min.)	Delayed Group	Time (min.)
Pretest	Individual	Program commenting and debugging tasks	varied	Program commenting and debugging tasks	varied
Intervention session 1	Grade-level pairs	Analyze WE set 1 Play *Coding Awbie*™	~10 ~10	Play *Coding Awbie*™	20
Intervention session 2	Grade-level pairs	Analyze WE set 2 Play *Coding Awbie*™	~10 ~10	Play *Coding Awbie*™	20
Intervention session 3	Grade-level pairs	Analyze WE set 3 Play *Coding Awbie*™	~10 ~10	Play *Coding Awbie*™	20
Midtest	Individual	Program commenting and debugging tasks	varied	Program commenting and debugging tasks	varied
Presentation	**Whole class**	**Programming applications**	**30**	**Programming applications**	**30**
Intervention session 4	Grade-level pairs	Play *Coding Awbie*™	20	Analyze WE set 1 Play *Coding Awbie*™	~10 ~10
Intervention session 5	Grade-level pairs	Play *Coding Awbie*™	20	Analyze WE set 2 Play *Coding Awbie*™	~10 ~10
Intervention session 6	Grade-level pairs	Play *Coding Awbie*™	20	Analyze WE set 3 Play *Coding Awbie*™	~10 ~10
Posttest	Individual	Program commenting and debugging tasks	varied	Program commenting and debugging tasks	varied

Note. WE stands for worked examples. The worked example sets are explained in Table 2.

Turn!, see Table 2, Set 1, Correct, for an example). Students wrote down the answers and solutions (or researchers wrote down their verbal answers) in response to these questions. The researchers did not provide feedback on whether students were correct or not, and sometimes the pairs did not agree.

Playing Coding Awbie

When pairs played *Coding Awbie*™, they started at the first level of the game and continued playing until their session time was up. If students finished a level, they moved on to the next level. We recorded their last completed level at the end of each session and started students on their next uncompleted level at the beginning of their next session. Therefore, students sometimes started over a level multiple times if they did not finish it during one or more of their sessions.

Test Item

During all three video-recorded testing sessions, we met with the students individually. Test items for the larger study evaluated students' programming conceptions and debugging and commenting in programming and mathematics. For this analysis, we focus on one, multi-solution programming item, which was the same across the pretest, midtest, and posttest. Students saw the image in Figure 1, without the grid markings and wrote a program using the *Coding Awbie*™ blocks to help Awbie collect the last strawberry (5E in Figure 1). Students ran their initial program and, if necessary, we gave them the opportunity to debug it. The item required Awbie to change directions and make jumps (two difficult practices identified in the first year) and allowed for students to repeat code (Figure 1, program C). This item provided rich data on students' algorithmic design accuracy as well as their debugging practices.

Table 2. The Complete Three Sets of Worked Examples Used in the Study

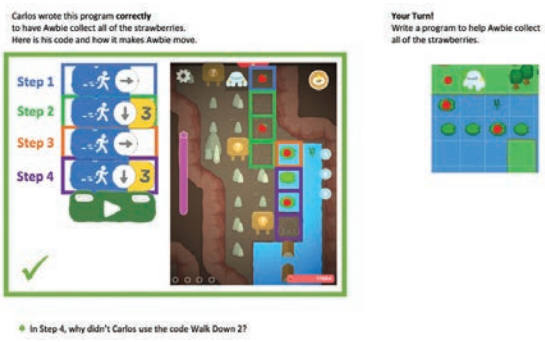
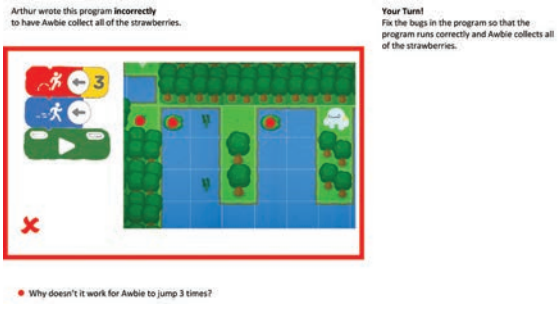
Set	Type	Worked Example	Pedagogical Design Choices
1	Correct		The first worked example in Set 1 showed color-coded steps to help students map each step of code with its corresponding movements in the game (supporting a trace the code strategy, see Table 3). This method of showing steps was too complicated to carry over into future examples, but it was important for helping them trace the code initially. In the first year of our study, we observed many students double-counting a square when Awbie switched directions (Kocabas et al., 2019). Therefore, this first example showed numbers being used correctly when Awbie switches directions. The *explanation prompt* regarding Carlos's method was meant to support accurate counting and to prompt students' wonderings about why Awbie might not be able to stop on a lilypad. Then students had to try programming Awbie on their own for a related situation.
1	Incorrect		The incorrect example in Set 1 involved an incorrect use of the jump, which drew students' attention to consider how far Awbie would move with each jump (i.e., jump over one space). This was a common challenge that students encountered in the first year of the study: determining how far Awbie would move when using the jump code (and realizing that Awbie will not keep jumping if he falls into the water). Then students had a chance to fix the program. A correct program would be jump left 1, walk left 1, jump left 1, walk left 1.

Table continued on next page

Table 2. The Complete Three Sets of Worked Examples Used in the Study *(continued)*

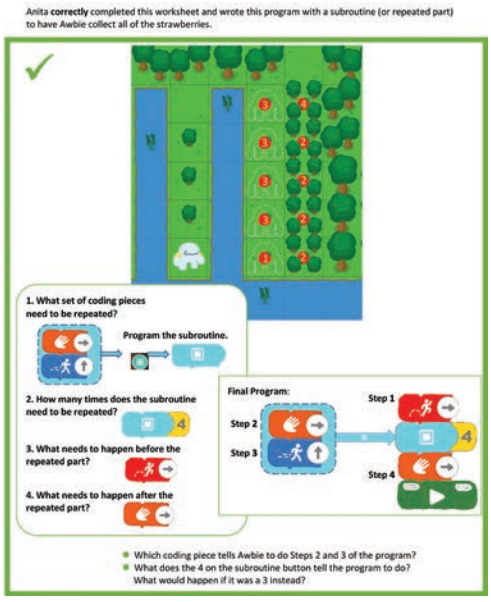
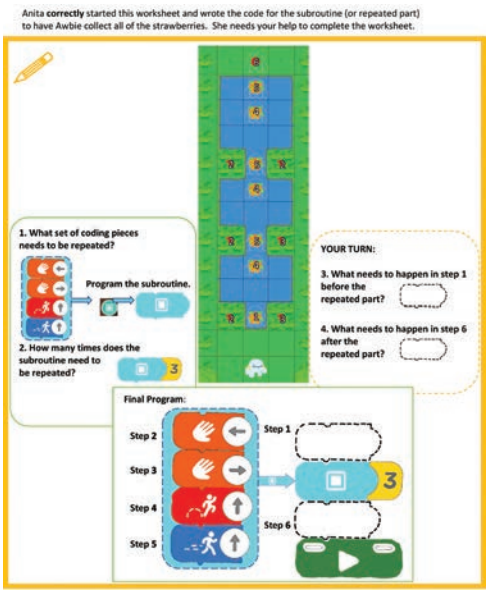
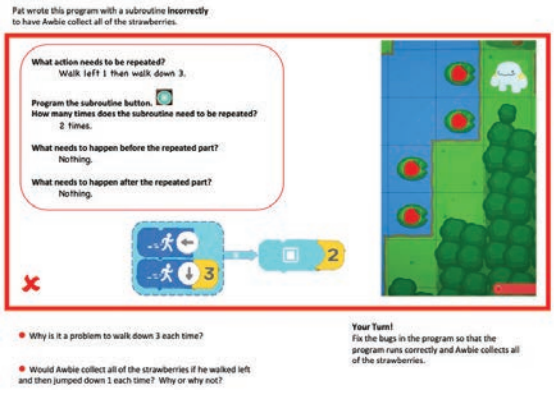
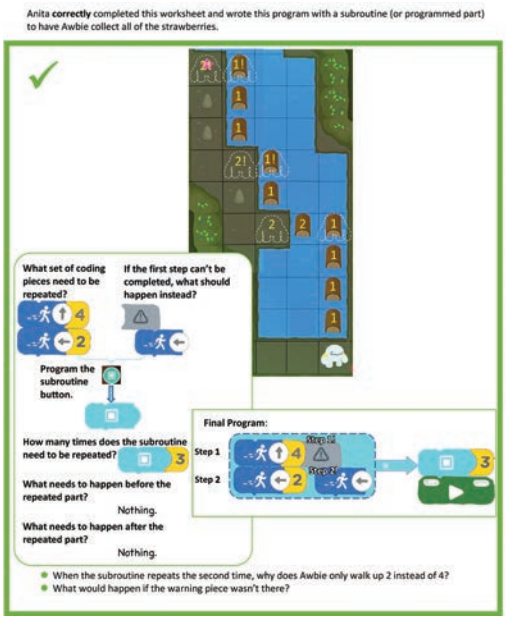
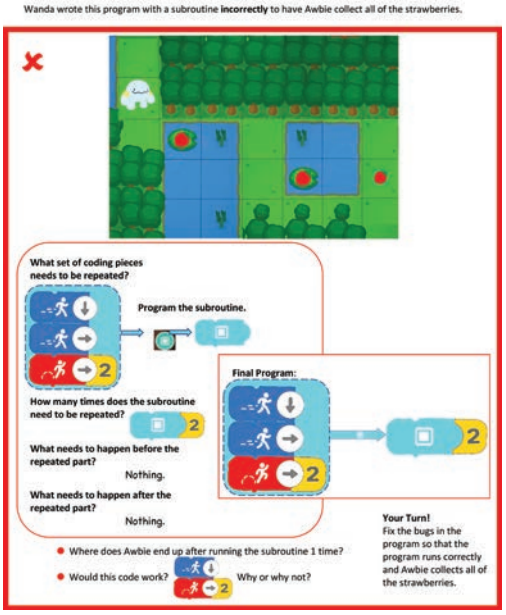
Set	Type	Worked Example	Pedagogical Design Choices
2	Correct		In the first year of the study, students expressed frustration with not being able to continue a program after using the loop command that is part of the game. Therefore, Set 2 of the worked examples first showed students how to correctly program a subroutine command to make a loop, as well as how to distinguish between walk and grab pieces. To help students understand that step 2 and step 3 were nested within the subroutine button, we showed this with the dotted box and arrow and also numbered the steps of the program in the picture to help students trace the code and see that Awbie repeats steps 2 and 3 four times. We also used *subgoal labels* in the form of a set of questions students could ask themselves to help explain the lines of code. The explanation prompts also helped students focus on the subroutine button and what the number on that button might mean.
2	Incomplete		To provide students with extra scaffolding on how the subroutine command works, we included an incomplete worked example in Set 2 where students completed missing code before and after a subroutine that operated as a loop. Therefore, students saw how the steps within the subroutine were repeated using the numbered steps in the picture but had to problem-solve to consider the missing steps. The incomplete worked example is a form of faded worked example, which is a fruitful method to use with older students in programming (Skudder & Luxton-Reilly, 2014) that we thought could work well with younger students as well.
2	Incorrect		The incorrect worked example in Set 2 required students to fix a double-counting bug within a subroutine. This worked example challenged students to combine their understanding of double-counting bugs and the use of the subroutine command. Further, the explanation prompt focused students' attention on the double-counting bug and encouraged them to consider another program that would get Awbie to the bottom space but without having him collect all of the strawberries (a goal of the program). Finally, students had a chance to fix the bugs.

Table continued on next page

Table 2. The Complete Three Sets of Worked Examples Used in the Study *(continued)*

Set	Type	Worked Example	Pedagogical Design Choices
3	Correct		Set 3 of worked examples started with a correct example that showed students how to use a warning (or conditional) command. In the first year of the study, students struggled with figuring out what the warning command does. Often, they placed it without any alternate code or in situations where it would not be needed. In fact, prior studies also found that children have difficulty learning the meaning of complex combinations of logic blocks through play alone (Wyeth, 2008), which further supported our use of a worked example for the warning command. Continuing to use the numbered steps, we used an exclamation point with a step number to show when the alternate, warning command code was used. Further, the explanation prompt encouraged students to make sense of why Awbie changes his number of steps he walks up.
3	Incorrect		Given the difficulty involved in using the warning command (and its relative infrequency of use in the game), we chose instead to include an additional incorrect worked example in Set 3 that required students to think about what was being repeated within a subroutine. This example also gave students another opportunity to consider how far Awbie would move when given the jump command.

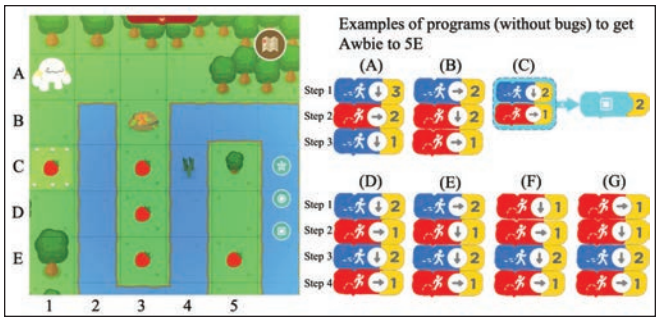


Figure 1. "Write a Program to Get Awbie to the Last Strawberry" (Point to 5E)

Analysis

First, if students' programs would get Awbie to 5E without bugs, we assigned their programs an accuracy score of 1; otherwise, we assigned them a score of 0. Second, we used Friedman's ANOVA to evaluate the effects of being in the immediate versus delayed groups across testing sessions on their initial programs' accuracy and used McNemar tests to further investigate the changes between testing sessions. Third, we analyzed students' programs for their types of initial bugs (e.g., wrong direction, hitting a rock, double-counting). Fourth, we

identified students' successful debugging strategies using Murphy et al.'s (2008) categories (see Table 3) along with one additional strategy we found. Lastly, we compared how many students per group correctly debugged their programs by creating accurate programs as described above.

RESULTS

Programming Accuracy

Based on a Friedman's ANOVA, the number of students who programmed Awbie to reach 5E significantly changed over the course of the study for students in the immediate group, $x^2(2) = 8.933$, $p = .011$, $r = .32$, and the delayed group, $x^2(2) = 8.133$, $p = .017$, $r = .38$. Based on follow-up McNemar tests, both groups made significant gains from pretest to posttest (immediate, gain of 26%, $p = .021$; delayed, gain of 36%, $p = .012$). However, only students in the immediate group significantly improved from pretest to midtest (gain of 35%, $p = .022$). Even more promising, nearly three-fourths of students in each group were able to correctly program Awbie on the posttest if given the chance to debug (see Table 4).

Programming Bugs and Debugging

The most prevalent bug students experienced that prevented them from getting Awbie to 5E was double-counting a square (See Figure 1). On the pretest, this happened in two ways. Six students double-counted the initial square (1A) and moved Awbie "walk right 3 (or miscounted 4)" in an attempt to get to 3A (or 4A) or "walk down 4 (or 5)" to get to 1D. Across all three tests, students also double-counted (see Table 5) by programming Awbie to walk down 3 (instead of walk down 2) for step

Table 3. Debugging Strategies

Strategy	Description
Tinkering[a]	Students appeared to randomly change out commands or numbers, or students changed multiple parts of the code at one time.
Reprogramming	Students removed all commands and numbers from their initial code and started over. Some students ignored the coding pieces they had used and started over with new ones. In other situations, students kept the coding pieces in the same order and moved them back into their workspace one by one as the student put together their new program.
Understanding the code[a]	Students tried to reason about what their code was doing and why there was a problem. For example, one student asked himself if using walk right would move Awbie one space.
Tracing the code[a]	Students moved their fingers along the path they expected Awbie to take to match each line of code. For example, if their first line of code told Awbie to walk right 2, they moved their finger two spaces to the right to trace where Awbie would move on the picture. On one occasion, a student used reprogramming but kept the coding pieces in the same order and used tracing at the same time to figure out where the bug was.
Pattern matching[a]	Students realized that something was not right or had a sense of where the problem was. Like subitizing where students see five objects and recognize it as five, some students saw the result of the code and just knew what needed to be changed. In many instances this happened when students saw Awbie moving too many spaces.

[a] See Murphy et al., 2008 for additional descriptions.

Table 4. Students Who Correctly Programmed Awbie Across Testing Sessions

Group	Before Debugging			After Debugging		
	Pretest	Midtest	Posttest	Pretest	Midtest	Posttest
Immediate (*n*=28)	18%	50%[a]	46%	32%	65%[a]	71%
First Grade (*n*=14)	0%	43%	36%	14%	57%	57%
Third Grade (*n*=14)	36%	58%[a]	57%	50%	75%[a]	86%
Delayed (*n*=27)	22%	37%	56%[b]	37%	59%	72%[b]
First Grade (*n*=14)	21%	43%	54%[b]	29%	50%	69%[b]
Third Grade (*n*=13)	23%	31%	58%[b]	46%	69%	75%[b]

[a] Two third graders did not take the midtest, so their data were not included.
[b] Two students (one first grade, one third grade) left before the posttest, so their data were not included.

three in Figure 1, programs D-G. When successfully debugging, students either used *pattern matching* and recognized that their "walk down 3" should be "walk down 2" or they *traced the code*, moving their finger along the path to show where Awbie would move, and noticed that they would only have to walk down 2.

Table 5. Percent of Students Double-Counting by Programming "Walk Down 3" Before Debugging

Group	Pretest	Midtest	Posttest
Immediate	14% (*n*=28)	8% (*n*=26)	21% (*n*=28)
Delayed	11% (*n*=27)	22% (*n*=27)	8% (*n*=25)

Another common bug in students' initial programs occurred because they did not realize that Awbie could not land on or walk through the rock in 3B. Ten students either started their programs with "walk down 1, jump right" or "walk right 2 (or jump right), walk down," which caused Awbie to bounce back to his previously originating square (1B or 3A). Six students fixed the bug when debugging, but one of these students made the same bug again on the midtest and posttest. One student's program only had this bug because she put her coding pieces together from bottom to top; she used an *understanding the code* strategy to realize that she needed to reorder her code.

Other less common bugs included moving Awbie in the wrong direction (i.e., up or left for their first line of code) or walking into water (e.g., moving Awbie from 1C to 3C by walking right instead of jumping). Sometimes students had a mismatch between what they said they were having Awbie do and what they programmed (e.g., said Awbie would jump right but programmed jump up); usually students caught these bugs when they saw the program run. A few students also ended their programs prematurely on 5D instead of 5E and were able to add on to their code when debugging.

Overall, students who successfully debugged their code primarily used *pattern matching* and *tracing the code* strategies, strategies which were embedded in the design of the worked examples and explanation prompts. Interestingly, students from the delayed group increasingly used a *reprogramming* strategy on the posttest. Based on the accuracy of students' initial programs, fewer students had to debug their programs from pretest to posttest, and for students who did debug, their debugging success rate, as shown by their making an accurate program, increased (i.e., more students successfully put together accurate programs after debugging on the posttest than on the pretest; see Table 6). Students who were not successful when debugging used a *tinkering* strategy and introduced a previously discussed bug or fixed an initial bug but did not fix a bug later in their code.

Table 6. Students Who Correctly Debugged Their Programs Across Testing Sessions

Correct for those who debugged			
Group	**Pretest**	**Midtest**	**Posttest**
Immediate	17% (*n*=23[a])	31% (*n*=13[a])	47% (*n*=15[a])
Delayed	19% (*n*=21[a])	35% (*n*=17[a])	36% (*n*=11[a])

[a] The number of students for each group and test who did not initially program Awbie correctly.

Discussion

Overall, this study adds to previous literature on the utility of worked examples (Joentausta & Hellas, 2018; Margulieux et al., 2016; Yan & Lavigne, 2014) and illustrates that early elementary students, who have little programming background, can benefit from analyzing correct and incorrect worked examples. Although both groups made significant gains after the six sessions in terms of accuracy of their programs, only students in the immediate group made significant gains from the pretest to the midtest. The immediate group's significant gains from pretest to midtest were largely due to the first graders' larger gains, providing further evidence that novices, even young ones, benefit from analyzing worked examples (Joentausta & Hellas, 2018), especially before playing.

Play Versus Worked Examples

More students (across both groups) double-counted after spending three sessions freely playing, and fewer students double-counted after analyzing worked examples. Students may have lost track of where Awbie would be as they put together the coding pieces when playing, increasing their likelihood to double-count a square. In fact, double-counting errors account for the immediate group's slight dip on the posttest in developing accurate programs before debugging. Further, on the posttest, when the immediate group was given the chance to debug and focus on their code like they did with the worked examples, they corrected their double-counting bugs (and third graders excelled); over half of these students used code tracing and pattern matching to debug, which are practices that were highlighted in the worked examples. Overall, both groups' accuracy increased when they were given the chance to debug. Unlike the immediate group, students in the delayed group made gains from just playing (from pretest to midtest); however, their gains were half as large as their peers in the immediate group. These results suggest that playing can help students improve, but incorporating worked examples with play provides added benefits, especially in terms of helping students debug. Students seemed to benefit from their playing the most initially (between pretest and midtest), which could account for why

the immediate group did not make gains between midtest and posttest unless they were given the opportunity to debug (which mimicked practices embedded in the worked examples). The delayed group continued to make gains when analyzing worked examples (from midtest to posttest), especially in the case of third graders.

Double-Counting Difficulties and Worked Examples

The results related to students' double-counting difficulties are a bit concerning, especially since the prevalence of the double-counting errors increased after playing the programming game. Game features may have contributed to this. For example, when Awbie moved too far and hit a tree in the game, he bounced back to the space he occupied before hitting the tree. If students programmed "jump right 2" to move Awbie in Figure 1 to 3A by misinterpreting the jump's movement, Awbie would hit the tree in 5A and bounce back to 3A. Therefore, students could reach their goal using these entertaining programs and may have learned that using numbers that are too large is not necessarily a problem, which could have contributed to their lax use of numbers on the midtest (for the delayed group) and on the posttest (for the immediate group). The worked examples helped lessen the double-counting bugs because the examples drew students' attention to the use of numbers and may have helped reorient their attention to the numbers. This result is impressive given that only one of the incorrect worked examples specifically focused on a double-counting bug, and it occurred within a subroutine (see Table 2: Set 2, Incorrect).

Worked Examples Supported Mental Representation in Debugging

Another benefit of analyzing worked examples, incorrect ones in particular, was helping students focus on debugging. Students in the delayed group transitioned from *tracing their code*, a popular strategy (Murphy et al., 2008), on the pretest to *reprogramming* their code on the posttest, but their reprogramming often involved them reusing pieces from their initial program and changing other pieces as needed. Thus, rather than focusing on following the path of their code with their fingers and changing specific lines of code, they followed the path mentally and rebuilt their program. These results provide some initial evidence that students may have made some progress in mentally representing the actions of the coding pieces. Students in the immediate group continued to use pattern matching, and although they also continued to trace their code, they did so with fewer hand movements, suggesting they were also mentally representing some of the movements.

Implications for Teachers

As teachers work to incorporate CT skills into their curriculum, our results suggest that having worked examples together with play experiences would be more advantageous than only having students play to learn basic programming and debugging. We suggest two instructional steps that teachers could leverage with students around early programming and debugging across programming environments: worked examples and code tracing.

Teachers could use worked examples as a pedagogical tool to draw students' attention to their own code, support all learners by building on their needs and strengths, and lessen performance gaps, as was the case for the first and third graders in the immediate group (i.e., the difference in their accuracy on the pretest was 36%, which reduced to 15% on the midtest). The prevalence and persistence of double-counting errors suggests that more worked examples and instruction around early programming and debugging should encourage discussing and exploring issues around counting in programming. Specifically, numbering or color-coding each step of the program in the worked examples could help students focus on the results of each movement and support students in tracing the code. Teachers should be careful to watch how students are using numbers and have discussions about programming for funny effects versus when precision is important. Teachers should use explanation prompts, such as the one posed about Carlos's method (see Table 2: Set 1, Correct), to help students focus on key aspects of code in programs. Further, teachers could present contrasting correct and incorrect worked examples to help students see the differences in coding with and without double-counting—which could prompt students to trace the differences between the code—and incorporate worked examples that target actions that are difficult to make sense of through playing only (e.g., subroutines; Kocabas & Bofferding, 2021) or common bugs.

Teachers could encourage students to trace their code by moving their finger along the path as they program to try and identify the code where the first bug happens and model this activity through whole group instruction. Encouraging students to reprogram mentally, teachers could then have students add each line of code while tracing the movements with their eyes, perhaps marking the beginning or ending points as needed for scaffolding. Over time, both reprogramming and tracing could lead to a more intentional pattern matching strategy. Before students reprogram or trace the code, teachers could have students articulate what the bug was and identify if it happened at the beginning, middle, or end of their code. Teachers could encourage students to become more precise about where the bug occurred within the code by asking questions such as, "Did the bug occur before or after Awbie jumped right?" Such questions, similar to the explanation prompts, could help students picture their code in parts and allow them to find a targeted spot in their code without tracing or reprogramming.

Acknowledgments

This research was supported by an NSF DRL ITEST Grant #1759254.

Correspondence concerning this work should be addressed to:

Laura Bofferding
Department of Curriculum and Instruction
100 N. University Street #4132
West Lafayette, IN, 47907
Email: *LBofferd@purdue.edu*

References

Atkinson, R. K., Derry, S. J., Renkl, A., & Wortham, D. (2000). Learning from examples: Instructional principles from the worked examples research. *Review of Educational Research, 70*(2), 181-214. https://doi.org/10.3102%2F00346543070002181

Elkin, M., Sullivan, A., & Bers, M. U. (2016). Programming with the KIBO robotics kit in preschool classrooms. *Computers in the Schools, 33*(3), 169-186. https://doi.org/10.1080/07380569.2016.1216251

Ginsburg, H. P. (2006). Mathematical play and playful mathematics: A guide for early education. In D. Singer, R. M. Golinkoff, & K. Hirsh-Pasek (Eds.), *Play = Learning: How play motivates and enhances children's cognitive and social-emotional growth* (pp. 145-165). Oxford University Press.

Griffin, J. M. (2016). Learning by taking apart: Deconstructing code by reading, tracing, and debugging. In *SIGITE '16: Proceedings of the 17th annual conference on information technology education* (pp.,148-153). https://doi.org/10.1145/2978192.2978231

Highfield, K. (2015). Stepping into STEM with young children: Simple robotics and programming as catalysts for early learning. In C. Donohue, *Technology and digital media in the early years: Tools for teaching and learning* (pp. 150-161). Routledge and the National Association for the Education of Young Children.

Ichinco, M., Harms, K. J., & Kelleher, C. (2017). Towards understanding successful novice example user in blocks-based programming. *Journal of Visual Languages and Sentient Systems, 3*, 101-118. https://doi.org/10.18293/vlss2017-012

Joentausta, J., & Hellas, A. (2018, February). Subgoal labeled worked examples in K-3 education. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education* (pp. 616-621). https://doi.org/10.1145/3159450.3159494

K-12 Computer Science Framework. (2016). Retrieved from http://wwwk12cs.org.

Kocabas, S., & Bofferding, L. (2021). Supporting third graders' use of subroutines in programming through play versus worked examples. In E. de Vries, Y. Hod, & J. Ahn, *Proceedings of the 15th International Conference of the Learning Sciences – ICLS 2021* (pp. 637-640). International Society of the Learning Sciences.

Kocabas, S., Bofferding, L., Aqazade, M., Haiduc, A., & Chen, L. (2019). Students' directional language and counting on a grid. In S. Otten, A. G. Candela, Z. de Araujo, C. Haines, & C. Munter (Eds.), *Proceedings of the 41st annual meeting of the North American Chapter of the International Group for the Psychology of Mathematics Education* (pp. 395-399). University of Missouri.

Margulieux, L. E., Catrambone, R., & Guzdial, M. (2016). Employing subgoals in computer programming education. *Computer Science Education, 26*(1), 44-67. http://dx.doi.org/10.1080/08993408.2016.1144429

McLemore, B., & Wehry, S. (2016). Robotics and programming in prekindergarten (RAPP): An innovative approach to introducing 4- and 5-year olds to robotics. *Global Learn 2016*. Association for the Advancement of Computing in Education. https://www.learntechlib.org/primary/p/172724/

Murphy, L., Lewandowski, G., McCauley, R., Simon, B., Thomas, L, & Zander, C. (2008). Debugging: The good, the bad, and the quirky – a qualitative analysis of novices' strategies. *SIGCSE '08* (pp. 163-167), Portland, OR: ACM. https://doi.org/10.1145/1352322.1352191

NGSS Lead States. (2013). *Next Generation Science Standards: For States, By States*. Washington, DC: The National Academies Press.

Parks, A. N., & Graue, B. (2015). *Exploring mathematics through play in the early childhood classroom*. Teachers College Press; National Council of Teachers of Mathematics.

Pérez, A. (2018). A framework for computational thinking dispositions in mathematics education. *Journal for Research in Mathematics Education, 49*(4), 424-461. https://doi.org/10.5951/jresematheduc.49.4.0424

Sipitakiat, A., & Nusen, N. (2012). Robo-blocks: Designing debugging abilities in a tangible programming system for early primary school children. *IDC 2012*, 98-105, ACM. https://doi.org/10.1145/2307096.2307108

Skudder, B., & Luxton-Reilly, A. (2014). Worked examples in computer science. *Proceedings of the Sixteenth Australasian Computing Education Conference*. Australian Computer Society, Inc.

Sullivan, A. A., & Bers, M. U., & Mihm, C. (2018). Dancing robots: integrating art, music, and robotics in Singapore's early childhood centers. *International Journal of Technology and Design Education, 28*, 325-346. https://doi.org/10.1007/s10798-017-9397-0.

Suzuki, H., & Kata, H. (1995). Interaction-level support for collaborative learning: AlgoBlock—an open programming language. In J. L. Schnase & E. L. Cunnius (Eds.), *Proceedings of CSCL '95: The first international conference on computer support for collaborative learning*. Bloomington, Indiana: Lawrence Erlbaum Associates. https://repository.isls.org//handle/1/4207

The AlgebraByExample Team. (n.d.) *AlgebraByExample's format*. SERP Institute. https://www.serpinstitute.org/algebra-by-example/format

Wang, D., Zhang, Y., & Chen, S. (2013). E-Block: A tangible programming tool with graphical blocks. *Mathematical Problems in Engineering*. DOI: 10.1155/2013/598547

Wing, J. M. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33-35. https://doi.org/10.1145/1118178.1118215

Wyeth, P. (2008). How young children learn to program with sensor, action, and logic blocks. *The Journal of the Learning Sciences, 17*(4), 517-550. https://www.jstor.org/stable/27736742

Wyeth, P., & Purchase, H. C. (2002). Tangible programming elements for young children. In *Proceedings of CHI 2002*, p. 774-755. https://doi.org/10.1145/506443.506591

Yadav, A., Hong, H., & Stephenson, C. (2016). Computation thinking for all: Pedagogical approaches to embedding 21st century problem solving in K-12 classrooms. *TechTrends, 60*, 565-568. https://doi.org/10.1007/s11528-016-0087-7

Yan, J., & Lavigne, N. C. (2014). Promoting college students' problem understanding using schema-emphasizing worked examples. *The Journal of Experimental Education, 82*(1), 74-102. DOI: 10.1080/00220973.2012.745466

Coding as Another Language: Computational Thinking, Robotics and Literacy in First and Second Grade

Marina Umaschi Bers, Madhu Govind, and Emily Relkin, *DevTech Research Group, Tufts University*

Corresponding Author: Marina Umaschi Bers, *marina.bers@tufts.edu*

Abstract

This paper explores the integration of coding, CT and literacy by describing a study conducted with first and second grade classrooms in Norfolk, Virginia. A total of 667 students and 57 educators from eight elementary schools, as well as 181 students from two comparison schools participated in a curriculum called Coding as Another Language (CAL) that utilizes KIBO robotics, a developmentally appropriate kit which does not require keyboards or screens. CAL positions the teaching of programming as a symbolic system of representation, a tool for creative expression and communication. Thus, research questions regarding the relationship between students' coding and CT outcomes and their literacy skills were explored, as well as teachers' reactions to the experience, in particular regarding the integrating of teaching computer science and literacy in the early grades. Participation in the CAL-KIBO curriculum was associated with improvement in coding and unplugged CT skills. Baseline literacy skills were related to students' acquisition of CT skills. For example, students who had higher literacy scores at the beginning of the term were more successful in CT tasks. Furthermore, although teachers varied in their perceptions of integrating coding and CT with literacy, our findings suggest that these disciplines may share some cognitive and pedagogical overlap that has yet to be extensively explored in the early computing education field.

Introduction

Educators, researchers, and policymakers are recognizing the need to give children access to computer science (CS) starting at an early age (Barron et al., 2011; Bers, 2018; Bers, 2019; Code.org, 2019; White House, 2016). Recently, the focus has expanded from teaching computer programming to also engaging with a set of underlying cognitive abilities known as computational thinking (CT) (Fayer, Lacey, & Watson, 2017; US Department of Education, Office of Educational Technology, 2017; Wing, 2006, 2011).

In an influential article entitled "Computational Thinking" that appeared in an issue of *Communications of the ACM*, Wing (2006) defined CT as "solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science" (p. 33). Wing argued that CT should be part of everyone's analytical repertoire. This echoed earlier work by Perlis (1962), who claimed that the "theory of computation" is for everyone, not only computer scientists, and Papert (1980), who proposed that through programming children can form ideas not only about computation but about thinking itself.

CT has received considerable attention over the past several years. There is consensus that CT must be available to thinkers of all disciplines, regardless of their ability to program (Guzdial, 2008; Yadav, 2016). However, there is little agreement on how to define it (Aho, 2012; Allan et al., 2010; Barr & Stephenson, 2011; Cuny, Snyder, & Wing, 2010; Grover & Pea, 2013; Lu & Fletcher, 2009; National Academies of Science, 2010; Relkin, 2018; Relkin & Bers, 2019; Shute, Sun, & Asbell-Clarke, 2017; Yadav, Good, Voogt, & Fisser, 2017). It is widely agreed that CT involves a broad set of analytic and problem-solving skills, dispositions, and habits, rooted in computer science but universally applicable. Examples include thinking recursively, using abstraction to identify salient pieces of a problem, and applying heuristic reasoning to discover a solution and/or identify potential "bugs" or problems (CSTA & ISTE, 2011; Kalelioğlu, Gülbahar, & Kukul, 2016). For definitions that are specifically relevant to young children, CT must also be framed in a developmentally appropriate context (Bers, 2018).

This research builds on these findings and focuses on the creative aspects of computer science for early elementary school children 5-9 years of age. We describe

a curriculum called Coding as Another Language (CAL) that focuses on the role of languages, both artificial and natural, for expressive purposes. CAL integrates the teaching of computer programming and literacy by positioning the teaching of CS as another medium for expression (Bers, 2019). In other words, the ultimate goal of mastering a programming language is not only to provide a means of problem-solving but also to allow creation of personally meaningful artifacts that can be shared with others. Ultimately, CAL is informed by the notion that both natural and artificial languages are symbolic systems of representation that can be used for creative expression and communication (Vee, 2017).

Prior research has shown that learning to code can enhance the acquisition of CT and related thinking abilities. Román-González et al. (2018) found improvements in CT in a study of middle school students (ages 12–14) who engaged in the code.org curriculum. Arfé et al. (2019) found improvements on neuropsychological tests of response inhibition and planning in first and second graders who received coding instruction. These studies provide evidence from randomized control trials that learning to code can accelerate the development thinking abilities critical to CT in children. Further studies are needed to evaluate the impact of learning to code with an integrated curriculum such as CAL on young children's CT skills and other aspects of their cognitive development.

We examined three different research questions: 1) How did the CAL curriculum promote students' coding and CT skills? 2) What was the relationship between students' CT skills and their literacy skills? 3) How did teachers react to the experience?

METHODS

Participants

A total of 667 first and second grade students and 57 educators from eight elementary schools (CAL group), as well as 181 students from two comparison schools (No-CAL group), participated in this study. All schools were in the Norfolk Public School district, Norfolk, VA, and participated in the CAL coding curriculum. The curriculum utilizes KIBO robotics, a developmentally appropriate kit designed for children 4 to 7 years old, which does not require keyboards or screens (Bers, 2018; Sullivan, Bers & Mihm, 2017; Sullivan, Elkin & Bers, 2015). The KIBO-21 kit used in this study consists of the KIBO robot, 21 colorful programming wood-based blocks, as well as light, distance, and sound modules and sensors. Children assemble the barcoded blocks, scan them using the robot's embedded barcode scanner, and press the triangle-shaped button on the robot to run the sequence of commands (see Figure 1).

Figure 1. KIBO-21 robotics kit

The CAL-KIBO Curriculum

The CAL-KIBO curriculum consists of 12-15 adaptable lessons administered over a 6-8-week period. Throughout the curriculum, children engage in activities, songs, games, and open-ended projects CAL-KIBO integrates coding and CT with the use of arts and crafts, reading and writing activities that are commonly used in early elementary school. For example, the final lessons involve a project based on a children's book, *Where the Wild Things Are* by Maurice Sendak. Students are invited to write a creative composition about what would happen in their own "Wild Rumpus" and subsequently program their "Wild Rumpus" using KIBOs (see Figure 2). The curriculum is aligned with the Common Core English Language Arts (ELA)/Literacy Framework, as well as Virginia CS Standards of Learning and other nationally recognized CS frameworks (e.g., K-12 CS Framework). Bers (2018) described seven powerful ideas of CT from CS that are developmentally appropriate for early childhood: hardware/software, algorithms, modularity, control structures, representation, debugging, and design process. Each CAL-KIBO lesson engages children in multiple powerful ideas of CT and connects them to foundational literacy and language concepts.

Figure 2. KIBO "Wild Rumpus" final projects created by second grade students

Procedure

Participating teachers received professional development prior to curriculum implementation, as well as ongoing professional learning consisting of virtual coaching and in-person support in the classroom provided by the district's instructional technologists. Teachers implemented the curriculum in their classrooms approximately twice a week (two one-hour lessons). At the end of each lesson, teachers completed a lesson log, a brief online survey that asked questions such as "What were some successes/challenges (if any) during this lesson?" and "Did you modify or adapt the activities in this lesson in any way?". Teachers were observed by the on-site project coordinator or instructional technologist at least two times over the course of the curriculum. Observers used the Positive Technological Development (PTD) Checklist (Bers, 2018) to examine how teachers and students were engaging with KIBO and with the CAL-KIBO lessons.

Measures

Over two years, multiple types of data were collected (see Table 1). Robotics mastery and CT development in both teachers and students were assessed before, during, and after the experience. CT assessments were administered to students who participated in the CAL-KIBO curriculum and to an age and demographically matched comparison group from two other schools in the same district. We collected and analyzed students' standardized literacy scores (DRA and PALS) from the beginning and end of the school year. DRA (Developmental Reading Assessment) is a computerized assessment that evaluates changes in K-8 students' reading level performance. PALS (Phonological Awareness Literacy Screening) is a diagnostic tool that measures children's developing word knowledge, oral reading in context, alphabetic, and phonemic awareness and is used to identify struggling readers and provide additional support.

To understand how teachers reacted to the experience of integrating coding and CT with literacy skills, we conducted surveys, interviews, and focus groups with participating teachers. Surveys were conducted before and after the professional development and consisted of questions related to teachers' self-perceptions of their general coding knowledge (e.g., "I know the definition of an algorithm"), pedagogical content knowledge surrounding how to teach coding (e.g., "I can teach lessons that integrate coding and literacy"), general KIBO robotics knowledge (e.g., "I can recognize common errors with the KIBO programming language and troubleshoot these errors"), knowledge of specific KIBO sensors and modules (e.g., "I know how to use KIBO's Sound Sensor"), attitudes and self-efficacy surrounding the implementation of the CAL-KIBO curriculum (e.g., "I am confident in my ability to implement the CAL-KIBO curriculum in my classroom") and perceptions on literacy (e.g., "What are your priorities in literacy instruction?"). *T*-tests were performed on pre and post-training survey data obtained from $n = 47$ participating first and second grade teachers.

Semi-structured interviews and focus groups with teachers and instructional technologists were conducted at various times (pre-training, during training, pre-curriculum, mid-curriculum, and post-curriculum) and focused on their reactions to KIBO and the CAL-KIBO curriculum. Examples of interview questions included "What has been easy/challenging? How do these activities fit into the rest of your classroom curriculum? If you were to use KIBO again in your classroom, how would you integrate it into your lesson plans?" In each interview, teachers were asked to reflect on their attitudes towards coding and robotics education, their perspectives on teaching literacy and the strengths and challenges of their experiences.

Measuring programming ability and CT skills in young children can be challenging. Two different tools were used to seek proof of learning of specific coding concepts through robotics. First, KIBO Mastery Challenges (KMCs), multiple-choice questions embedded in the curriculum, were administered after specific lessons, and a composite score was calculated from difficulty indices (Hassenfeld et al., 2020). Second, TACTIC-KIBO, a summative assessment of coding and CT skills, was administered after participation in the curriculum. Because children worked in teams throughout the curriculum, KMCs and TACTIC-KIBO provided individualized data regarding learning outcomes that would go unnoticed by just looking at students' final projects. In addition to tool-specific assessments, a validated "unplugged" CT assessment called *TechCheck* was used, which focuses on problem-solving skills of

Table 1. Research Questions and Data Analysis Plan

Research Question	Data Sources	Data Analysis Method
RQ 1: How did the CAL curriculum promote students' coding and CT skills?	TACTIC-KIBO, KIBO Mastery Challenges (KMCs), TechCheck	Descriptive, correlation, t-tests
RQ 2: What was the relationship between students' coding and CT skills and their literacy skills?	KMCs, TechCheck, PALS, DRA	Descriptive and correlation analyses
RQ3: How did teachers react to the experience?	Teacher interviews, surveys, and lesson logs	Thematic analysis, t-tests

the kind required to carry out computer programming without requiring knowledge or experience with coding (Relkin et al., 2020). *TechCheck* requires the transfer of knowledge gained from coding into CT skills useful for solving unplugged challenges that are not explicitly taught in the CAL-KIBO curriculum. *TechCheck* was administered before and after the curriculum in both intervention and control groups (see Table 2). Only results from neurotypical students are included in the analyses that follow since the CT and coding assessment measures have yet to be validated with a neuro-diverse population.

Data Analysis

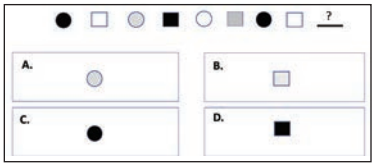
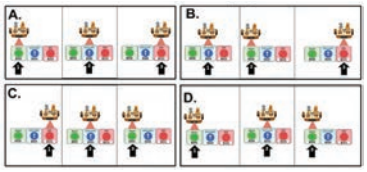
Paired sample *t*-tests and generalized linear mixed modeling were performed on assessment results from students who received CAL-KIBO, as well as students who participated in non-coding classroom activities, to address how the curriculum impacted students' CT skills. To address the relationship between students'CT skills and literacy, correlation, regression, and Bayesian mixed effect modeling were conducted on data from a subgroup of n = 191 students from among the total sample of N = 667 participants representing those who completed pre and post *TechCheck* as well as the DRA and PALS literacy measures.

Teacher interviews and focus groups were transcribed and then analyzed using Braun and Clarke's (2006) six-phase thematic analysis approach. This approach involved reading through the data multiple times, generating initial codes, combining codes into overarching themes, exploring how the themes connected to our initial research question, refining themes with greater detail, and drafting our findings while referring to the data to ensure that our findings provided an accurate representation of teachers' experiences of the curriculum. Common trends derived from this thematic analysis are presented in this paper.

RESULTS

This section organizes the findings based on the three research questions addressed by the study: the CAL curriculum impact on coding and CT skills, the relationship between coding and CT with literacy, and teachers' reactions.

Table 2. Child Study Measures

Measure	*TechCheck*	TACTIC-KIBO	KMCs
Reference	Relkin, de Ruiter, & Bers (2020)	Relkin (2018)	Hassenfeld et al. (2020); Relkin & Bers (2020)
Description	Assessment using "unplugged" (non-coding) tasks to measure CT related problem-solving abilities	Assessment of platform-specific coding and CT abilities in seven sub-domains.	Formative assessment of programming concepts specific to the CAL-KIBO curriculum as "checks of learning". Assesses understanding of semantics and syntax of programming without requiring them to solve problems
Example	What comes next? 	What is the correct order to scan program blocks? 	Which block makes KIBO shake? 
Specifications	• 15 multiple-choice questions • Designed for children ages 5-9 • 15 minutes to administer • Score range 0-15 • Validated against expert assessment of CT abilities	• 28 multiple-choice questions • Designed for ages 5-9 • 30-40 minutes to administer • Score range 0-28 • Validated against expert assessment of CT abilities	• 4 assessments, 6 multiple-choice questions each totaling 24 questions • Multiple-choice format • Designed for children ages 5-9 • High interrater reliability • Difficulty index for each question calculated with more weight given to difficult questions. Questions summed into a weighted Difficulty Composite Score

CAL Curriculum Impact on Students' Coding and CT Skills

Descriptive statistics for all coding and CT assessments are shown in Table 3. First and second grade students who participated in the CAL-KIBO curriculum improved significantly on the *TechCheck* assessment, $t(666)= 10.55$, $p < .001$. A grade-matched control group that participated in non-coding classroom activities, the control group, did not significantly improve, $t(180) = 1.81$, $p = .07$. The improvement after 6-8 weeks of CAL-KIBO instruction is consistent with the estimated change in baseline *TechCheck* scores in the absence of coding instruction over approximately six months. A Generalized Linear Mixed Model incorporating taking into account age, grade, classroom, gender, and baseline score revealed that exposure to the CAL-KIBO curriculum was a significant predictor of the *TechCheck* outcome scores, $p < .01$ (Relkin et al., 2021).

We conducted stratified analyses to look for effects by grade. First grade students who received CAL-KIBO improved significantly on *TechCheck*, $t(270) = 9.21$, $p < 0.001$, whereas the control group did not, $t(358.31) = 1.07$, $p = .95$. Second graders in the CAL group also improved significantly, $t(395) = 6.11$, $p < 0.001$ but not as much as first graders possibly due to a ceiling effect on the *TechCheck* assessment in which high baseline scores in second graders reduced the window for observing change (Relkin et al., 2020). A more challenging version of the assessment for second graders has since been created to address this issue (Relkin et al., 2021). A smaller but borderline significant improvement was observed in the second grade non-coding group, $t(109) = 2.34$, $p = .05$ possibly due to a learning effect or chance. Results stratified by grade shows that although first and second graders both improved on *TechCheck* more than their non-coding counterparts, we observed more of a difference in first graders.

Relationships Among Students' Coding, CT, and Literacy Skills

The theoretical framework upon which the CAL curriculum is designed proposes that learning computer programming allows children to gain an alternative form of literacy that permits self-expression in ways that are similar to reading and writing (Bers, 2019; Vee, 2017). Thus, in this study we examined the correlations between coding, CT, and conventional measures of literacy. Specifically, we wanted to know if students who scored higher on state-wide assessments of literacy also performed higher in coding and CT tasks after they completed the CAL-KIBO curriculum. Our measure of coding ability was the student's KMC composite score, and our measure of CT was the student's *TechCheck* score post-curriculum. To estimate students' literacy skills before the intervention, we used Fall standardized literacy scores (DRA and PALS) obtained by the school. There was a moderate positive correlation (Pearson) between baseline *TechCheck* scores and both the DRA ($r = 0.39$, $p < .0001$) and PALS ($r = 0.33$, $p < .0001$). We carried out linear regression to examine whether these two baseline literacy measures predicted the endpoint *TechCheck* scores when baseline *TechCheck* performance was taken into account. A model containing baseline PALS and *TechCheck* scores significantly predicted end point *TechCheck* $F(2, 190) = 38.47$, $p <.0001$. A model containing baseline DRA and *TechCheck* scores also significantly predicted end point *TechCheck* $F(2, 190) =36.18$, $p <.0001$. We conducted Bayesian mixed-effects modeling with *TechCheck* scores as the outcome variable and PALS and DRA literacy measures as well as baseline *TechCheck* scores as predictors. The Bayes factor was >100 indicating "decisive evidence" that baseline literacy measures (PALS and DRA) were predictors of end point *TechCheck* score when baseline *TechCheck* score was included in the model. In summary, these analyses indicate a possible relationship

Table 3. Descriptive Statistics for Coding and CT Variables

	n	Mean (SD)	Min	Max
***TechCheck* Baseline CAL Group**[a]	667	10.09 (2.61)	3	15
***TechCheck* End Point CAL Group**	667	11.03 (2.55)	2	15
***TechCheck* Baseline No-CAL Group**	181	9.50 (2.38)	3	14
***TechCheck* End Point No-CAL Group**	181	9.77 (2.55)	4	14
***TACTIC-KIBO* First Grade**[b]	214	13.10 (3.33)	2	20
***TACTIC-KIBO* Second Grade**	398	18.28 (3.90)	4	26
***KMCs* Second Grade**[c]	217	3.44(1.05)	1.59	6.25

Note.
[a] *TechCheck* assesses children's unplugged problem solving and CT
[b] The Tufts Assessment of Computational Thinking in Children - KIBO version (*TACTIC-KIBO*) assesses children's platform specific coding and CT skills
[c] *KMCs* (KIBO Mastery Challenges) assesses children's KIBO coding proficiency

between the acquisition of CT skills and baseline literacy skills. This could be a consequence of the measures reflecting a child's developmental stage or literacy skills influencing assessment performance on *TechCheck* rather than a specific effect on learning CT.

Teachers' Reactions

Most teachers expressed a high level of engagement when first introduced to KIBO during the training and displayed excitement while working on their own robotic projects. Results of *t*-tests indicated statistically significant increases in each of the 27 survey items assessing $n = 47$ teachers' knowledge of general coding concepts, KIBO skills, and CS pedagogy as well as their attitudes towards coding and robotics education, *t*'s ranging from 5.19-22.40, $p < .001$ across all t-tests. Across all survey items and domains, neither race/ethnicity nor teaching experience impacted participant responses.

We present our qualitative findings from teachers in Table 4, which summarizes teachers' perceived successes and challenges of their curriculum experience, reactions to the coding-literacy integration, and overall factors that impacted curricular implementation. In terms of the CAL curriculum, teachers' interviews and surveys showed that teachers enjoyed KIBO and their students did as well. However, the organization of robotics materials presented its own set of challenges, particularly with shifting materials between classrooms and managing clean-up time. Teachers developed different strategies such as creating a rotational system with fellow teachers, selecting students to be in charge of robotics clean-up, and keeping a set of 3-4 KIBOs in their classrooms at all times. Despite these logistical challenges, during interviews, teachers described being drawn to the hands-on, tangible nature of KIBO, especially in comparison to other screen-based applications felt. Teachers felt KIBO was engaging and developmentally appropiorate for their students.

Table 4. Teachers' Reflections of the CAL-KIBO Curriculum Experience

Topic	Theme	Illustrative Quote
Successes and challenges	Coding beyond the screen	"I learned that coding doesn't just involve sitting in front of a computer and typing things and that it actually involves just using your mind and talking things out and stuff like that"
	KIBO organization	"Because I was sharing with [name omitted] and others that only did it once a week, so [the KIBOs] were in and out, and things got mixed up, so I color-coded mine"
Coding-literacy integration	Resistant to integration	"I almost think they should take the writing components out of it, and just let us focus on the actual straight coding."
	Neither resistant nor receptive	"I'm not reinforcing, 'Oh, capitalization, grammar, this and that', like that's just not happening...I do feel like it hit, definitely, on oral communication... They have to communicate with their buddy, whoever they're working with"
	Receptive to integration	"Each day with each lesson the kids were writing...and reading different things. They had to read [it] over. They had to read other people's instructions."
Factors impacting curricular implementation	Time spent preparing for and implementing lessons	"Most of the lessons are supposed to be an hour, I know, but mine were probably two or more... I was able to tie [the Design Journals] in with the writing more because I spent more time on it."
	Teacher collaboration and utilization of resources	"One of our teachers broke down the KIBO [lesson] for the day and made a PowerPoint, so that we would be able to follow through and...check off the steps as we did them."
	Competing priorities of other lessons and activities	"Teachers are always pressed with a pretty comprehensive curriculum, so adding this in addition was a little overwhelming at times."
	Classroom management	"I felt even groups of four would be too much of a chaotic ruckus. And I felt like the kids would be most successful if they're just working in a partnership."
	Flexibility in adapting lesson activities	"I know everybody adapted and I adapted it by adding extra time. But I stuck to the curriculum. I know some people kinda cut out certain things and whatever, but I wanted to give them the full experience, so I pretty much went by the book."

Teachers varied in how they responded to the integration of literacy in the CAL curriculum; however, there was a distinct trend amongst second grade teachers. During interviews and focus groups, it became clear that teachers who understood literacy instruction as singularly focused on discrete skills (e.g., phonics, punctuation, etc.) were less open to the CAL curriculum and to the overall integration of CT, robotics, and literacy. Conversely, teachers who understood literacy in broader terms and saw meta-cognitive ideas and concepts about reading and writing as essential to the development of robust literacy abilities (e.g., communication, creative expression, awareness of audience and purpose, etc.) were more open to the curriculum.

The analysis of teacher interviews and focus groups revealed several factors that impacted teachers' overall experience: time spent preparing and implementing lessons; collaboration and utilization of resources; competing priorities of other lessons and activities; classroom management; and flexibility in adapting lesson activities. Individual classroom and school contexts played an important role. For instance, teachers who were more successful with the curriculum had manageable classroom sizes, flexible schedules to accommodate CAL lessons, and an adequate number of robotic kits for students to work in small groups. Conversely, teachers who had a large number of students with little floor space, taught in an open-classroom setting, or had rigid grade-level schedules faced more challenges.

Discussion

Participation in the CAL-KIBO curriculum was associated with improvement in coding and unplugged CT skills. It is noteworthy that the measure of unplugged CT (*TechCheck*) showed improvement with exposure to CAL-KIBO even though the curriculum did not explicitly include the types of unplugged activities in *TechCheck*. This finding supports the assertion that the problem-solving improvements were a consequence of knowledge and skills gained while learning to code and not a function of explicit instruction in solving unplugged challenges.

Taken together, our analysis suggests that baseline literacy skills were related to students' acquisition of CT skills. Students who had higher PALS or DRA scores at the beginning of the term were more successful in CT tasks measured by *TechCheck*. These findings may help us develop effective integrated CS curricula and identify core skills that need to be strengthened so that all students can reap the benefits of early childhood computer education.

We encouraged teachers to adapt the curriculum based upon their students' needs and available time. As a consequence, there was variability in the fidelity of implementation of the CAL-KIBO curriculum across schools and classrooms. Other sources of variation included having a different number of robots available in each school. The classroom dynamic differed when KIBO robots were shared by pairs of students as opposed to larger groups of 5-6 students. Classroom size and space were other variables that impacted the classroom experience. The CAL-KIBO curriculum was successful in engaging teachers and generating a high level of enthusiasm. Limitations on time to implement the curriculum and difficulty in organizing materials were common challenges reported by educators.

Implications

The key to successful educational initiatives is to make authentic connections to the teaching that is already going on in the classroom. In this study, we connected coding and CT to literacy. Our findings indicate that first and second grade students improved in their coding and CT skills as a result of participating in the CAL-KIBO curriculum. Although teachers varied in their perceptions of integrating coding and CT with literacy, our findings suggest that these disciplines may share some cognitive and pedagogical overlap that has yet to be extensively explored in the early computing education field. This integration can have a positive impact regarding learning outcomes. In addition, children who participated in the CAL-KIBO curriculum did better on unplugged CT challenges (*TechCheck*) than their counterparts. This improvement occurred despite unplugged CT challenges not being an explicit part of the CAL-KIBO curriculum, suggesting that a transfer of knowledge took place.

Based on our study, we provide the following recommendations for practitioners seeking to integrate CT and coding in their classrooms:

- Time: Allocate enough time in the weekly schedule to prepare for and implement the curriculum. Implementing the curriculum in the winter or spring enables teachers to utilize established classroom routines and behavioral expectations, which are key to maximizing young student's engagement and learning.
- Curricular Alignment: Although the CAL-KIBO curriculum focused primarily on the connections between literacy and CS, and was taught during the literacy block, teachers found ways to connect the activities to other curricular domains such as science and math. We recommend aligning with multiple subject areas, not just literacy, while still framing the teaching of coding as a form of creative expression and communication. The more teachers could see how the lesson aligned with other content instruction, the more they felt comfortable teaching.
- Resources: Teachers benefitted from collaborating with one another, using the virtual and face-to-face support resources, having access to knowledgeable support staff, and co-teaching lessons with instructional technologists. These findings provide insight into the

types of professional learning that early elementary teachers may require to feel comfortable and confident when teaching coding and robotics.

The CAL-KIBO curriculum focused on a single robotics coding platform, KIBO. We have also developed a version of the CAL curriculum that utilizes the free ScratchJr introductory programming language and are currently conducting studies. Future work will address the relative strengths and weaknesses of both curricula regarding different coding platforms and develop collaborations with other school districts in the U.S. and abroad. We will also explore if the CT skills acquired through one programming language transfers to another one.

References

Aho, A. V. (2012). Computation and computational thinking. *The Computer Journal, 55*(7), 832–835. https://doi.org/10.1093comjnl/bxs074

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? *Inroads, 2*(1), 48–54. https://doi.org/10.1145/1929887.1929905

Barron, B., Cayton-Hodges, G., Bofferding, L., Copple, C., Darling-Hammond, L., & Levine, M. (2011). Take a giant step: a blueprint for teaching children in a digital age. New York: The Joan Ganz Cooney Center at Sesame Workshop. https://joanganzcooneycenter.org

Bell, T., & Vahrenhold, J. (2018). CS unplugged—how is it used, and does it work?. In *Adventures between lower bounds and higher altitudes* (pp. 497–521). Springer, Cham. https://doi.org/10.1007/ 978-3-319-98355-4_29

Bers, M. U. (2018). *Coding as a Playground: Programming and Computational Thinking in the Early Childhood Classroom*. New York, NY: Routledge Press.

Bers, M. U. (2019). Coding as another language: a pedagogical approach for teaching computer science in early childhood. *Journal of Computers in Education, 6*(4), 499-528.

Braun, V., & Clarke, V. (2006). Using thematic analysis in psychology. *Qualitative Research in Psychology, 3*(2), 77–101. https://doi.org/10.1191/1478088706qp063oa

Code.org (2019). https://code.org/

Core Team, R. (2019). *R: a language and environment for statistical computing*. Vienna: R Foundation for Statistical Computing. https://www.R-project.org/

ISTE, CSTA (2011). *Operational definition of computational thinking for K–12 education*. https://id.iste.org/docs/ct-documents/computational-thinking-operational-definition-flyer.pdf

Cuny, J., Snyder, L., & Wing, J. M. (2010). Demystifying computational thinking for non-computer scientists. Unpublished manuscript in progress, referenced in https://www.cs.cmu.edu/~CompThink/resources/TheLinkWing.pdf

Fayer, S., Lacey, A., & Watson, A. (2017). *BLS spotlight on statistics: STEM occupations-past, present, and future*. Washington, D.C.: U.S. Department of Labor, Bureau of Labor Statistics. https://www.bls.gov

Grover, S., & Pea, R. (2013). Computational thinking in K–12: a review of the state of the field. *Educational Research, 42*(1), 38–43. https://doi.org/10.3102/0013189X12463051

Hassenfeld, Z. R., Govind, M., de Ruiter, L. E., & Bers, M. U. (2020). If You Can Program, You Can Write: Learning Introductory Programming Across Literacy Levels. *Journal of Information Technology Education: Research*, 19, 65-85. https://doi.org/10.28945/4509

Hermans, F., & Aivaloglou, E. (2017). To Scratch or not to Scratch? A controlled experiment comparing plugged first and unplugged first programming lessons. WIPSCE 2017. *Proceedings of the 12th workshop in primary and secondary computing education* (pp. 49–56).

Kalelioğlu, F., Gülbahar, Y., & Kukul, V. (2016). *A framework for computational thinking based on a systematic research review*. https://www.researchgate.net/publication/303943002_A_Framework_for_Computational_Thinking_Based_ on_a_Systematic_Research_Review

Lu, J. J., & Fletcher, G. H. (2009). Thinking about computational thinking. In *ACM SIGCSE Bulletin* (Vol. 41, No. 1, pp. 260-264). ACM. https://doi.org/10.1145/1539024.1508959

Relkin, E. (2018). Assessing young children's computational thinking abilities (Master's thesis). Retrieved from ProQuest Dissertations and Theses database. (UMI No. 10813994).

Relkin, E. & Bers, M. U. (2019). Designing an Assessment of Computational Thinking Abilities for Young Children. In L.E. Cohen & S. Waite-Stupiansky (Eds.), *STEM for Early Childhood Learners: How Science, Technology, Engineering and Mathematics Strengthen Learning* (pp. 85-98). New York, NY: Routledge.

Relkin, E., de Ruiter, L., & Bers, M. U. (2020). TechCheck: Development and Validation of an Unplugged Assessment of Computational Thinking in Early Childhood Education. *Journal of Science Education and Technology*. https://doi.org10.1007/s10956-020-09831-x

Relkin, E., de Ruiter, L., & Bers, M. U. (2021). Learning to Code and the Acquisition of Computational Thinking by Young Children. *Computers & Education*.

Rodriguez, B., Rader, C., & Camp, T. (2016). Using student performance to assess CS unplugged activities in a classroom environment. In *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education* (pp. 95-100). ACM. https://doi.org/10.1145/2899415.2899465

Shute, V. J., Sun, C., & Asbell-Clarke, J. (2017). Demystifying computational thinking. *Educational Research Review*, 22, 142–158. https://doi.org/10.1016/j.edurev.2017.09.003

Sullivan, A., Bers, M. U., & Mihm, C. (2017). Imagining, Playing, & Coding with KIBO: Using KIBO Robotics to Foster Computational Thinking in Young Children. *Proceedings of the International Conference on Computational Thinking Education*. Wanchai, Hong Kong.

Sullivan, A., Elkin, M., & Bers, M. U. (2015). KIBO Robot Demo: Engaging young children in programming and engineering: *Proceedings of the 14th International Conference on Interaction Design and Children (IDC '15)*, Medford, MA, June 21-25. New York, NY: ACM

Thies, R., & Vahrenhold, J. (2013). On plugging "unplugged" into CS classes. *Proceeding of the 44th ACM technical symposium on Computer science education (SIGCSE '13)*. Association for Computing Machinery, New York, NY, USA, 365–370. https://doi.org/10.1145/2445196.2445303

U.S. Department of Education, Office of Educational Technology (2017). Reimagining the role of technology in education: 2017 National Education Technology Plan update. https://tech. ed.gov/teacherprep

Vee, A. (2017). *Coding Literacy: How Computer Programming Is Changing Writing*. Cambridge, MA: The MIT Press. https://mitpress.mit.edu/books/coding-literacy

Virginia Department of Education (2021). 2016 Virginia Acts of Assembly Item 138, page 117. https://www.doe.virginia.gov/instruction/computer-science/2016-acts-of-assembly-page117.pdf

White House (2016). *Educate to innovate*. https://www.whitehouse.gov/issues/education/k-12/educate-innovate

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

Wing, J. (2011). *Research notebook: computational thinking—What and why?* The Link Magazine, Spring. Carnegie Mellon University, Pittsburgh. https://www.cs.cmu.edu/link/researchnotebookcomputational-thinking-what-and-why

Yadav, A., Hong, H., & Stephenson, C. (2016). Computational thinking for all: Pedagogical approaches to embedding a 21st century problem solving in K-12 classrooms. *TechTrends 60*, 565-568. DOI: 10.1007/s11528-016-0087-7.

Yadav, A., Good, J., Voogt, J., & Fisser, P. (2017). Computational thinking as an emerging competence domain. In *Technical and vocational education and training* (Vol. 23, pp. 1051–1067). https://doi.org/10.1007/978-3-319-41713-4_49

Sphero.Math: A Computational Thinking-Enhanced Fourth Grade Mathematics Curriculum

David Weintrop, Janet Walkoe, Margaret Walton, Janet Bih, Peter Moon, Andrew Elby,
College of Education, University of Maryland,
Bianca Bennett, and Madison Kantzer, *District of Columbia Public Schools*

Corresponding Author: David Weintrop, *weintrop@umd.edu*

Abstract

Computational thinking (CT) constitutes an essential set of skills and practices that all students should learn in order to effectively and meaningfully participate in an increasingly computational world. This paper introduces Sphero.Math, a curriculum that integrates CT concepts and practices into 4th-grade mathematics classrooms using the Sphero robot. Sphero.Math was co-designed with school district partners in such a way as to achieve two central design goals. First, integrate CT and mathematics in a mutually-supportive way, meaning that learners engage with CT as a means to deepen mathematics learning and that mathematics serves as a context to learn CT. Second, create a CT-infused curriculum using existing school/district resources that is designed to fit within the constraints of public school classrooms. Along with introducing Sphero.Math and its design goals, this paper presents empirical evidence for the types of CT and math learning opportunities that Sphero.Math can provide for students. In doing so, this work contributes to our understanding of ways to productively integrate CT into elementary classrooms and advances our understanding of how to work within existing educational infrastructure to provide effective and equitable CT learning opportunities for all students.

Introduction

Computing, and the technologies it enables, are playing an increasingly important role in society. As such, for young learners growing up in this technological landscape, being able to recognize the capabilities and limitations of computing technologies, think critically about the roles computing plays in society, and most crucially, to be able to meaningfully participate in a technological culture is essential. In response to the growing importance of the skills associated with computing and the need to broaden participation in the community, it is essential that all students have the opportunity to learn the big ideas of computing and develop foundational computational thinking (CT) skills (National Research Council, 2010; Shute et al., 2017; Wing, 2006).

To ensure equitable access to CT learning opportunities, CT must be a part of all learners' K-12 classroom experiences. To accomplish this goal, we have developed a curriculum, entitled Sphero.Math, in which learners explore mathematical concepts and develop CT skills by interacting with a spherical robot. The approach embeds CT into existing 4th-grade math classrooms and has both theoretical and practical motivations. From a theoretical perspective, there is a long history of research demonstrating the mutually-supportive potential of computing and mathematics (e.g., Abelson & diSessa, 1986; Kaput et al., 2002; Noss & Hoyles, 1996; Papert, 1972). This research shows how computing can serve as a tool for deep mathematical exploration and that mathematics as a subject can provide a meaningful context to enact computational ideas. From a practical perspective, taking this integrative approach ensures that all learners will have access to CT learning experiences as every school has resources to teach mathematics (e.g., teachers, time, classrooms) and every student takes mathematics in 4th-grade. Further, the Sphero.Math curriculum was co-developed with teachers and district curriculum experts as part of a research-practice partnership (Coburn & Penuel, 2016), resulting in significant institutional support and a curriculum that fits with the technology, infrastructure, and professional development resources available within the district.

The last decade has seen a flourishing of research seeking to bring CT into STEM classrooms of all levels. In their review of assessing CT, Tang and colleagues

(2020) identified 96 empirical studies of CT, with roughly two-thirds of those studies (67.4%) focused on formal education and 21.7% of those studies investigating CT integrated with STEM content. The Sphero.Math project adds to the growing body of research investigating CT in elementary classrooms (e.g., Asbell-Clarke et al., 2020; Israel & Lash, 2019; Miller et al., 2020) and adds to it in a number of unique ways, including its explicit focus on mutual-supportiveness (i.e., a focus on mathematical and CT learning), its close collaboration with the school district to ensure ease of adoption and continued district support, and its focus on broadening participation and equity by working in schools that historically have offered few CT learning opportunities.

While there remains an active discussion as to what exactly constitutes CT (Grover & Pea, 2013; Shute et al., 2017), given our focus on CT in elementary mathematics classrooms, our conceptualization of CT draws from the CT in Math and Science Taxonomy (Weintrop et al., 2016) and how it aligns CT to the unique characteristics of the disciplines. More concretely, we operationalize CT using the PRADA (pattern recognition, abstraction, decomposition, and algorithms) model for integrated CT (Dong et al., 2019) along with CT practices associated with programming, including iterative development and debugging.

This paper continues with an introduction to the Sphero.Math curriculum and a discussion of the context in which the study took place and the methodological approach used. After that, we present our results in the form of two vignettes, showing what the Sphero.Math curriculum can look like in practice and how learners have opportunities to engage in both mathematics and CT in mutually-supportive ways. These vignettes are intended to serve as an existence proof that CT and mathematics can co-exist and be embedded into elementary mathematics classrooms in a way that is consistent with the goals of the teacher and district while also engaging learners with CT concepts historically absent from such learning contexts. The paper concludes by summarizing the significance and implications of this work. Collectively, this work advances our understanding of how to work within existing educational infrastructure to bring high-quality CT instruction into urban elementary classrooms. Further, this work contributes an empirical example of how to integrate CT into elementary classrooms in an equitable, sustainable, mutually-supportive way.

METHODS

In this section, we first introduce the Sphero.Math curriculum, providing a high-level description of the approach and technology to help situate the specific examples provided in the Results section. We then describe the study design and data collection strategies followed by a presentation of the setting in which this research occurred and the characteristics of the participants.

Sphero.Math

The Sphero.Math curriculum consists of 14 lessons that engage students in solving math tasks using a Sphero (Figure 1a), which is a spherical robot that can be programmatically controlled with a tablet or smartphone (Figure 1b). Sphero programs take the form of block-based scripts (Figure 1c) that can include basic programming constructs (e.g., loops, conditionals), commands to control the Sphero's movement and appearance (e.g., `roll`, `strobe`), and sensor data from the device (e.g., distance traveled, speed). The basic movement command for the Sphero is the roll command which takes three inputs: heading, speed, and time, resulting in a command that reads: `roll 45° at 100 speed for 3s`. The decision to use the Sphero for this curriculum was made by the district, which had classroom sets of Spheros and tablets available for students. The Sphero robot and its accompanying programming environment have a number of affordances that lend it well to the exploration of mathematical concepts, including defining distance as a product of rate and time (thus supporting proportional reasoning), recording and reporting distance traveled, speed, and acceleration for a given program run that can then be recorded and analyzed, and interactive input features for expressing mathematical concepts, such as an interactive protractor to define angles (Figure 1d). Additionally, the physical nature of the Sphero supports younger learners in drawing on their bodies

(a) (b) (c) (d)

Figure 1. (a) The Sphero robot, (b) programming environment, (c) a sample program, and (d) interactive protractor for defining angles.

and experiences moving through the world as a means of translating their intentions into programming commands (Bih et al., 2021).

Sphero.Math was developed collaboratively by researchers from the project working closely with a technology specialist, an instructional coach, and a 4th-grade teacher from the partnering district. Lessons were designed to fit within the time allotted for mathematics instruction. Each lesson follows a three-part structure: *Engage* - which situates the lesson's central problem/question, *Code* - during which students author, test, and debug programs, and *Debrief* - during which student review their programs, reflect on the process, and discuss the mathematical and computing concepts encountered during the lesson. Each lesson has both student and teacher-facing materials, with the teacher-facing materials defining a clear objective for the lesson and documenting how the lesson aligns to the 4th-grade Common Core Mathematics Standards (CCSS, 2010), which are the standards used by the district. The lessons also detail what CT students will engage with during the lesson, and, when possible, what lessons from the district-mandated mathematics curriculum the Sphero.Math lesson aligns with. In an effort to provide a meaningful and engaging context, the curriculum situates each activity in an amusement park theme.

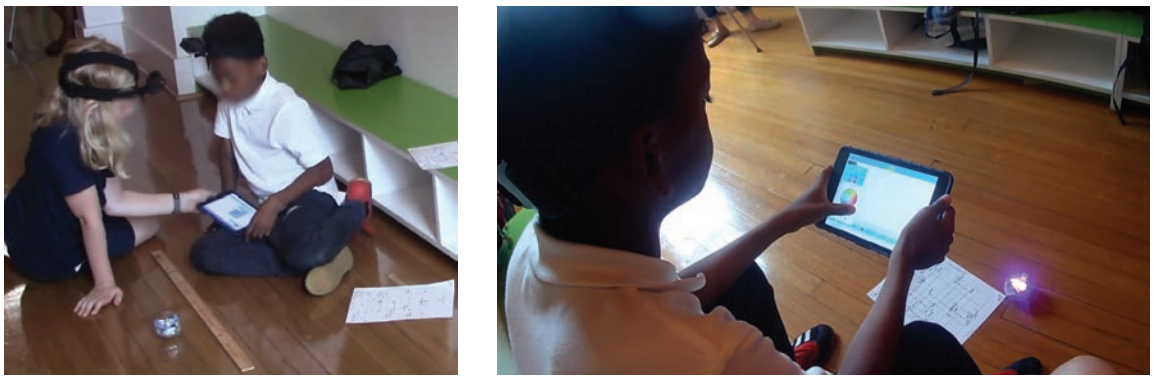
Central to each lesson is the integration of both mathematical and CT concepts. For example, in one lesson entitled *Roller Coaster Speed Test*, students calculate the expected distance that the Sphero should travel at speeds of 10 and 100 for 1, 2, 3, and 4 seconds. The students then compared their calculated distance to the actual distance that the Sphero travels for the different speeds and times. During the lesson, students examine relationships between factors of 10, which is a 4th-grade Common Core Standard (CCSS.Math.Content.4.NBTA.1). They also attend to issues like mathematical precision, which is Practice 6 of the Common Core Standards for Mathematical Practice (CCSS, 2010), and the various factors that can affect the Sphero's precision (like rolling on tile vs. rolling on carpet). Also, while not part of the 4th-grade math curriculum, students gain insight into covariation (how variables affect each other as they change) as they work to understand how changing the speed and time variables affects distance. At the same time, students develop CT skills, like developing algorithms, as they write a program that commands Sphero to roll at a certain speed for a certain amount of time, and manipulate those variables in the program as necessary.

Setting and Participants

This study took place over two years in 4th-grade mathematics classrooms in an urban public school district in the Mid-Atlantic region of the United States. The Sphero.Math curriculum has been taught in schools across the district that serve a range of student populations. In Year 2 (the 2020-21 school year), we focused on two schools, one that serves a predominantly Black student population (99% Black student body) and a second that serves a predominantly Hispanic/Latino student population (74% Hispanic/Latino; 12% White; 11% Black), with both schools being designated as Title I by the district, meaning they serve a significant number of students from economically disadvantaged households. The vignettes presented below are from a recorded session with a pair of students in Year 1, which is a more racially diverse school (51% White, 17% Hispanic/Latinx, 15% Black/African American, 9% Asian, 8% Mixed Race).

Data Collection

To understand the experiences of students working through the Sphero.Math activities, we conducted a qualitative classroom study that closely examined how a small group of students engaged with the Sphero.Math curriculum. In Year 1, we observed Sphero.Math lessons being taught in person, and recorded video of students working through Sphero.Math activities. For each classroom observation, two focal pairs were identified by the classroom teacher.



(a) (b)

Figure 2. Students working on a Sphero.Math activity as seen from (a) the stationary camera and (b) a head-mounted camera.

Each focal student wore a head-mounted video camera, providing a first-person perspective on their experience working with the Sphero and the Sphero programming environment (Figure 2b). Additionally, a third, stationary camera was used to provide a third-person perspective on the pairs' engagement with the materials (Figure 2a). This resulted in three videos for each pair of students working through the lesson. At the conclusion of the Sphero.Math lessons, brief interviews were conducted with the students asking them to reflect on their experiences going through the lesson. We did not have the same opportunity to video record students in Year 2 due to the COVID-19 pandemic. Instead, Sphero.Math lessons were taught virtually by the classroom teachers and observed by researchers.

Data Analysis

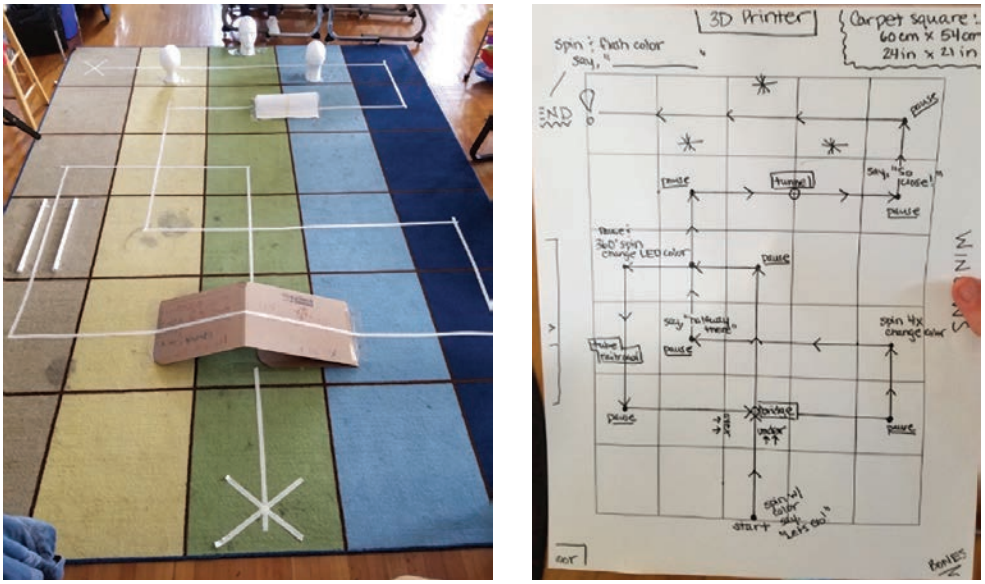
The data presented in this paper is from the video-recorded sessions of student pairs in Year 1. Two members of the research team watched and thematically coded all three videos of each Sphero.Math lesson (Saldaña, 2015). In this initial viewing, researchers were specifically looking for moments where CT and mathematics were co-expressed and mutually-informing (i.e. CT was being used to investigate a mathematical concept or mathematics was being used to explain a CT idea or outcome). These moments of co-expression were then transcribed and analyzed by the full team using interaction analysis techniques (Jordan & Henderson, 1995), looking to identify the mathematical content and the types of CT and programming skills exhibited by the students. As we were interested in identifying features of the tools and tasks that helped support these moments of mutually-supportive CT and mathematics. The research team watched the video selections together multiple times until consensus was reached on the mathematical ideas and CT used by the students. In analyzing these instances of co-expression, we matched students' mathematical skills to the Common Core State Standards (CCSS, 2010). In terms of CT, we used the PRADA framework (Dong et al., 2019) as a lens to identify the different components of CT that students used in conjunction with the mathematics. In addition, we looked for several key programming skills, including iteratively developing a solution and debugging, which are generally considered to be important elements of CT (Shute et al., 2017). Table 1 shows the CT components we focused on, how they were operationalized for this work, and examples of each from the Sphero.Math curriculum.

RESULTS

In this section, we present two vignettes of one pair of students (shown in Figure 2) working through a summative Sphero.Math activity. In the first year of the project, the summative Sphero.Math activity asked students to program their Sphero to navigate a maze (Figure 3a). However, unlike a conventional maze-navigation challenge, students were

Table 1. The definition of computational thinking used in this work.

CT Component	Definition	Sphero.Math Example
Pattern Recognition	Identifying parts of a problem that repeat. Repeating steps/concepts can often be performed by a computer	Seeing that a series of steps are repeated to measure a distance, so putting those steps inside a looping block (e.g. repeat)
Abstraction	Creating generalizations that are meaningful/useful or identify the essential parts of a problem and re-representing the problem including only the relevant parts	Being given the challenge of finding the perimeter of a specific rectangle but then writing a program that can calculate the perimeter of any rectangle
Decomposition	Breaking a problem down into smaller parts (either sub-problems or specific aspects of a problem that can be easily mapped to a computational solution)	Breaking down a complex shape into a series of line segments and turns which map easily onto programming commands far a Sphero
Algorithms	Defining a sequence of steps that can be followed to achieve some desired outcome	Defining a series of steps that can be executed by the Sphero to find the area of your classroom
Programming Skills	**Definition**	**Sphero.Math Example**
Iterative development	Developing a solution through incremental steps, iteratively refining and revising aspects of the solution	Manipulating the duration argument in the roll command multiple times to "zero in" on a time that will make Sphero roll the desired distance
Debugging	A systematic approach to identifying the source of undesired outcomes and correcting them to achieve desired results	Reading through Sphero code line-by-line and acting out the steps to identify the source of an error



(a) (b)

Figure 3. (a) The maze to be completed and (b) the map of the maze, include the dimension so the carpet squares (provided inches and centimeters).

not allowed to test out their program on the maze directly, instead, they were provided a map of the maze (Figure 3b). On the map, the path of the maze is overlaid onto a grid with the dimensions of the grid provided. To navigate the maze, students must calculate the length of maze segments based on the provided grid-square dimensions and then program their Sphero to travel that distance. Only after the students had written the program to traverse the maze based on their own measurements and calculations were they able to try it out on the actual maze. The activity aligns with 4th-grade Common Core Mathematics Standards (CCMS) asking learners to "Solve problems involving measurement and conversion of measurements" and the CT practices of Pattern Recognition, Algorithms, Decomposition, along with foundational programming practices. These vignettes are intended to demonstrate the ways that CT and mathematics can be mutually-supportive and highlight the role of the Sphero in mediating this co-expression. After each vignette, we present a brief discussion highlighting the mathematics and CT demonstrated by the students and how the two practices are mutually-supportive.

Decomposition, Iterative Development, and Debugging

At the outset of the activities, the two boys in our focal pair identify the first segment of the maze as consisting of three-and-a-half squares and initially develop a plan to write a program that has their Sphero travel the length of one carpet square and then plan to reuse that command throughout the maze. The decision to program the Sphero to travel just 1 square length is in contrast to trying to write a program that travels the full distance of the initial segment (3.5 squares). This is significant as it demonstrates the students setting out to solve the maze-navigation problem by decomposing the maze into reusable segments, creating a basic command that travels one square, and then reusing it to navigate the maze

To get their Sphero to roll one carpet square length (24 inches as defined by the map), the students start by composing a program consisting of the following command: `roll 0° at 50 speed for.7s.` They then place their Sphero next to a yardstick and run the program (shown in Figure 2a). After their first run, the Sphero rolled 32 inches. Realizing their Sphero rolled too far, they modify their program, instructing the Sphero to only roll for .3 seconds. They re-run their program but this time the Sphero only travels 6 inches. They next try .5 seconds and re-run the program again, this time the Sphero travels 23.5 inches, prompting Student One to say *"I think [the teacher] would give us that".* Student Two replies, *"No, because if it kept going like that…"* then trails off as his partner says, *"Ok then .573".* They modify the program, changing the third argument in the roll command to .573 seconds, and run the program for a fourth time. The Sphero travels exactly 24 inches, prompting Student Two to say, *"Right on the dot."*

Decomposition, Iterative Development, and Debugging: Discussion

In the first vignette, we see two 4th-grade students decomposing the maze into segments (CT Practice:

Decomposition) and then attending to issues of precision (CCSS Mathematical Practice 6). In their efforts to program their Sphero to travel exactly one grid square, the students employ a strategy of trying out a duration, recording the resulting traveled distance, and then revising the duration based on the outcome (CT Practice: Iterative Development). While students used an iterative approach, they also demonstrated an understanding of place value and reasoned about the size of numbers expressed in decimal form as they chose .573 for their duration that is both greater than .5 and less than .7, the previously attempted values. This decimal comparison is a 4th-grade Common Core Standard (CCSS.Math.Content.4.NF.C.7) and aligns to concepts the 4th-graders learned during their traditional math lessons. Also, while not in the curriculum, this lesson also allows students to explore covariation, as they change the time the Sphero travels to achieve the desired distance.

A second instance of CT can be seen in the final comments from this vignette. In debating whether a "close enough" program would work, Student Two comments: *"No, because if it kept going like that…"* This utterance demonstrates rather sophisticated reasoning about the implications of iteration, grounded in an understanding of measurement and additive properties of length. Because the pair plans to re-use this program for each square of the maze, being off by a little will produce a compounding error where their Sphero ends up increasingly off the more times this command is executed. In this utterance, we can see how the context of programming the Sphero serves as a context to reason about and engage with various aspects of measurement and precision, which are part of the 4th-grade math standards.

Proportional Reasoning, Pattern Recognition, and Algorithms

Having successfully authored a program to get their Sphero to travel one square, the pair returns to the map to figure out how to get the Sphero to travel the entirety of the first maze segment. Pointing to the start position, Student One says *"alright, we've done one of them, so we'll do one more".* Student Two picks up this idea, *"so then we need to go right here and right here, and then we need to go half of one"* moving his finger along the line segment as he speaks. After his finger reaches the end of the first segment, Student Two says *"So we'll just cut the speed in half"*, proposing a solution to traveling that last half-square. To implement this plan, the pair repeats the command they figured out in the previous vignette (`roll 0° at 50 speed for.573s`) three times, once each for the full squares of the first maze segment. For the final command, they cut the speed in half, having the Sphero travel at speed 25 instead of speed 50. The resulting program to navigate the first segment of the maze is showing in Figure 1c.

Proportional Reasoning, Pattern Recognition, and Algorithms: Discussion

In this second vignette, the students start to build their algorithm based on the command they authored in the first vignette to travel a single map square. They identify that their one-square program needs to be repeated 3.5 times to traverse the first segment of the maze (CT Practice: Pattern Recognition). To do so, they develop an algorithm that repeats the full square command three times and then they devise a fourth command to travel half of a map square by cutting the speed in half (CT Practice: Algorithms). At the same time, students work with a unit of measurement (a grid square) and determine how to manipulate the measurement to solve a distance problem (Standard: CCSS.MATH.CONTENT.4.MD.A.2). Additionally, realizing that cutting the speed in half while keeping the duration fixed will result in the Sphero traveling half-as-far shows these 4th-grade students conducting multiplicative comparisons to solve problems (Standard: CCSS.Math.Content.4.OA.A.1) and employing proportional reasoning, which is not expected of students until grade 6.

Discussion

Across these two brief vignettes, we see instances of students engaging in mathematical thinking and CT practices and also ways that two are mutually informing: the mathematics task is serving as a context for employing CT practices and that CT practices serve as a way to support and enact mathematical reasoning. One important thing to note across these vignettes is that the CT practices and mathematical concepts demonstrated in these vignettes were all enacted in service of solving a specific problem. The assignment was not for students to show an understanding of a particular concept, nor did they receive direct instruction on how to use, for example, proportional reasoning, to navigate the maze or iterative development to refine the precision of their program. Instead, these practices and concepts emerged in situ, informed by the specific constraints of the program and affordances of the technology.

The final noteworthy aspect of this work is to reiterate the context in which it occurred - an urban elementary mathematics classroom. As a result of the way the curriculum was designed, through an RPP including input from teachers and district administrators, the curriculum relied only on materials available to teachers in the district and aligned to mathematics standards the students are evaluated against. Additionally, the district provided time for teacher professional development and helped recruit schools and teachers to participate in the project. Collectively, the alignment of the goals of the teacher, the district, and the researchers led to the creation of a curriculum that attends to both CT learning and

mathematical learning and does so in a way that allows for it to be taught in classrooms that historically have had little CT or technological components.

Implications, Limitations & Future Work

There are several implications of this work for both researchers and practitioners. Central among them is a demonstration of the potential for mutually-supportiveness between CT and disciplinary content - in this case, 4th-grade mathematics. The vignettes above show that CT-enhanced activities can provide a context for students to engage in mathematical problem solving and that math problems, like manipulating variables to achieve a particular distance, can provide a context for CT learning. This is significant as it directly addresses a central (and valid) concern of many educators - that there is no room for additional content in the already overpacked school day (Barr & Stephenson, 2011). It is important to mention that bringing a CT-infused curriculum into classrooms still requires an investment of time and energy (e.g., professional development to train teachers, district logistics support to ensure materials are present in each classroom). However, when the commitment to bring CT to all students is in place, a mutually-supportive curriculum can provide a pathway to achieve this goal.

The second implication of this work is a reimagining of what, how, and where students can engage with CT. Historically, the concepts and practices associated with CT (e.g., debugging, problem decomposition, programming) have resided in high school computer science classes. With this work, we show what it can look like for these same ideas to reside in a very different educational context: elementary mathematics classrooms. This serves as a demonstration of how an expanded view of CT can be integrated into existing subjects, opening new pathways for learners, especially younger students, to be introduced to the powerful ideas of computing.

The data presented in this study are intended to serve as an existence proof that it is possible to work within the constraints of urban, public schools to provide CT learning opportunities to young students in a way that aligns with the disciplinary goals of the class. Given the nature of the data collected (multi-stream video of a small number of students), this work cannot make claims about the impact of this work across the full set of students or other schools that were using the Sphero.Math curriculum. This larger-scale analysis is a limitation of the present study and is the planned course of future work. Our intention with this work is to lay the qualitative foundations for the quantitative evaluation work to follow.

In closing, this work shows how an integrated approach, co-developed with district partners, can bring CT learning opportunities to students who historically have had few opportunities to engage with computing. In this case, teaching CT in elementary schools that serve economically disadvantaged student populations composed of learners from populations historically excluded from computing. This is a direct result of the close collaboration with the district and reflects the district's commitment to providing enriching CT activities to all students across the district. The Sphero.Math project was conceived through conversations with district leadership and co-designed with teachers, enabling the voices of both researchers and practitioners to be present at each phase of the work. The result is a curriculum that aligns with the district's mission of equitable and joyful learning opportunities for all learners and delivers on the overarching goals of bringing CT to all learners.

Acknowledgments

This work is supported by the Spencer Foundation (#201900099). However, any opinions, findings, conclusions, and/or recommendations are those of the investigators and do not necessarily reflect the views of the Foundation.

References

Abelson, H., & diSessa, A. A. (1986). *Turtle geometry: The computer as a medium for exploring mathematics*. The MIT Press.

Asbell-Clarke, J., Rowe, E., Almeda, V., Edwards, T., Bardar, E., Gasca, S., Baker, R. S., & Scruggs, R. (2020). The development of students' computational thinking practices in elementary- and middle-school classes using the learning game, Zoombinis. *Computers in Human Behavior*, 106587.

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.

Bih, J., Weintrop, D., Moon, P., & Elby, A. (2021). Computational Bodies: Grounding Computational Thinking Practices in Embodied Gesture. *Proceedings of the 15th International Conference of the Learning Sciences*, 171–178.

Coburn, C. E., & Penuel, W. R. (2016). Research–Practice Partnerships in Education: Outcomes, Dynamics, and Open Questions. *Educational Researcher, 45*(1), 48–54.

Dong, Y., Catete, V., Jocius, R., Lytle, N., Barnes, T., Albert, J., Joshi, D., Robinson, R., & Andrews, A. (2019). PRADA: A Practical Model for Integrating Computational Thinking in K-12 Education. *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, 906–912.

Grover, S., & Pea, R. (2013). Computational Thinking in K-12: A Review of the State of the Field. *Educational Researcher, 42*(1), 38–43.

Israel, M., & Lash, T. (2019). From classroom lessons to exploratory learning progressions: Mathematics + computational thinking. *Interactive Learning Environments, 0*(0), 1–21.

Jordan, B., & Henderson, A. (1995). Interaction analysis: Foundations and practice. *The Journal of the Learning Sciences, 4*(1), 39–103.

Kaput, J., Noss, R., & Hoyles, C. (2002). Developing new notations for a learnable mathematics in the computational era. *Handbook of International Research in Mathematics Education*, 51–75.

Miller, E. C., Severance, S., & Krajcik, J. (2020). Connecting Computational Thinking and Science in a US Elementary Classroom. In J. Anderson & Y. Li (Eds.), I*ntegrated Approaches to STEM Education: An International Perspective* (pp. 185–204). Springer International Publishing.

National Governors Association Center for Best Practices, Council of Chief State School Officers. (2010). *Common Core State Standards for Mathematics*. National Governors Association Center for Best Practices, Council of Chief State School Officers.

National Research Council. (2010). *Report of a Workshop on The Scope and Nature of Computational Thinking*. The National Academies Press.

Noss, R., & Hoyles, C. (1996). *Windows on mathematical meanings: Learning cultures and computers*. Kluwer.

Papert, S. (1972). Teaching Children to be Mathematicians Versus Teaching About Mathematics. *International Journal of Mathematical Education in Science and Technology, 3*(3), 249–262.

Saldaña, J. (2015). *The coding manual for qualitative researchers*. Sage.

Shute, V. J., Sun, C., & Asbell-Clarke, J. (2017). Demystifying computational thinking. *Educational Research Review, 22*, 142–158.

Tang, X., Yin, Y., Lin, Q., Hadad, R., & Zhai, X. (2020). Assessing computational thinking: A systematic review of empirical studies. *Computers & Education*, 103798.

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

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

Improving Teacher Use of Educational Robotics To Teach Computer Science in K-5 Mathematics

Theodore J. Kopcha, Cheryl Y. Wilson, and Dayae Yang, *The University of Georgia*

Corresponding Author: Theodore J. Kopcha, *tjkopcha@uga.edu*

Abstract

This paper reports on a professional development (PD) effort in which 12 elementary teachers (K-5) in a rural, under-resourced school were equipped with the skills needed to integrate computer science activity into the regular math curriculum. Over the course of a year, the teachers engaged in a week-long summer workshop and monthly follow-up training with in-classroom support as they integrated educational robots into their math curriculum. Surveys, tests, and teacher lesson plans were analyzed to better understand how the PD contributed to teacher's knowledge and skills. Wilcoxon Signed-Rank tests indicated that teacher rankings on their attitudes towards and planning practices with educational robots were statistically significantly higher at the end of the PD. The improvement was further reflected in the quality of their lesson plans. Implications for preparing teachers to use educational robots to teach computer science in elementary mathematics are discussed.

Educational robotics is a popular way to introduce young children to computer science and the mathematics inherent in computer programming (Anwar et al., 2019; Bers, 2010). Papert's (1980) *Mindstorms* famously captures the inherent potential of educational robotics that is present today — when children work with robots, they are challenged to think in ways that are analogous to the computer. They must visualize the robot's movements, program the robot to make those movements, and continue to test and debug until they achieve their goals (Anwar et al., 2019; Yuen et al., 2015). These activities position educational robotics as a way of meeting the current national priority of improving CS education for all in the US.

The process of problem solving inherent in educational robotics is commonly referred to as computational thinking. Computational thinking is a problem-solving process associated with computer scientists in which children use computational practices to solve problems (Wing, 2011). It consists of skills like decomposing a larger task into subgoals, using and designing algorithms to achieve goals with precision (i.e., algorithmic thinking), recognizing and forming patterns, and iteratively developing a computational solution through trial and error (Grover & Pea, 2013; Wing, 2006).

Codable robots provide opportunities to apply computational thinking practices within the context of mathematical instruction. For example, when children repeatedly test and debug a computer program, they are making sense of problems and showing perseverance in solving them (Dunbar & Rich, 2020). Programming the robot requires children to move between fractional and decimal forms of numbers, including whole numbers, as they determine the speed and time needed to move the robot (i.e., algorithmic thinking). The robot's movement can support learning the relationship between different angle measures (Dunbar & Rich, 2020), the fundamental components of coordinate systems (Rich et al., 2020), and even proportional reasoning between distance, speed, and time (Rich et al., 2020; Yuen et al., 2015). Thus, educational robots can provide opportunities to integrate CT within K-12 classrooms and support children's mathematical learning (Grover & Pea, 2013; National Science & Technology Council, 2018).

While educational robotics provides an opportunity to integrate computer science into the math curriculum, elementary teachers often lack the support needed to do so successfully. Research has indicated an overall lack of access to quality computer science activities and the professional learning needed to develop teacher's knowledge and skills with activities like educational robotics (Anwar et al., 2019; Yadav et al., 2016; Yuan et al., 2019). This often leads teachers to view educational robotics as an additional burden that detracts from meeting the standards to which they are held accountable (Ketelhut et al., 2020). Thus, the question remains how to support elementary teachers to integrate CT through educational robotics with their math instruction (Lee et al., 2017; Leonard et al., 2018).

Professional development (PD) can equip teachers with the knowledge and skills needed to integrate educational robotics into the elementary curriculum while meeting content area standards (Ketelhut et al., 2020; Yuan et al., 2019). This is important for achieving the national vision of CS for all students, particularly in under-resourced communities. However, improving a teacher's attitude toward and use of educational robotics is particularly important. As elementary teachers increase their use of educational robotics, students have opportunities to develop foundational computational thinking skills at a young age. Positive exposure to CT and computer programming at a young age can help children view themselves as capable of and interested in a career in computer science (Eguchi, 2014; Karp & Maloney, 2013; Yuan et al., 2019). Given the demand for CS education in today's schools, there is a need for research on specific approaches that can support teacher integration of CS through educational robotics (Anwar et al., 2019; Ketelhut et al., 2020).

This paper reports a PD effort in which 12 elementary teachers (K-5) in a rural, under-resourced area learned to integrate educational robotics into the regular math curriculum. The year-long PD blended research-based principles of technology integration PD (e.g., Mouza, 2009; Kopcha, 2012) with Smith & Stein's (2011) five practices for orchestrating mathematical discourse to support teachers in meeting state standards while engaging students in educational robotics. Survey data and lesson plans were analyzed to understand how the PD contributed to the teachers' learning and success. The questions guiding this study were:

A. What is the influence of PD on elementary teachers' attitudes toward and knowledge of teaching math through educational robotics?
B. How did the elementary teachers integrate educational robotics to support Smith and Stein's (2011) mathematical practices?

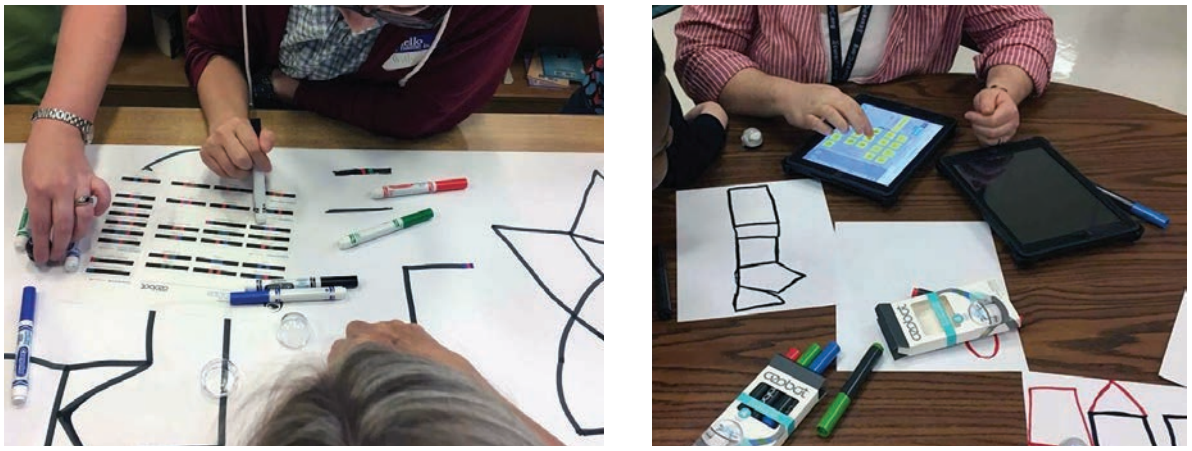
Professional Development Framework

Our professional development (PD) focused on the knowledge, skills, and attitudes needed to integrate educational robotics into the mathematics curriculum. We used Ozobot robots because this technology supports learning CS concepts from Kindergarten through fifth grade. As shown in Figure 1, Ozobots can be programmed visually using hand-drawn commands such as following solid lines and performing actions through color combinations (e.g., red-green-red = go faster). Ozobots can also be programmed to move, change speed, turn, and perform actions (e.g., zig zag; change color) through block-based programming (see Figure 2). For younger learners, commands are represented visually (e.g., a 90° turn as a circle with one quarter filled in) whereas more advanced learners can use text-based blocks to enter specific values.

To address both the technological skills and teaching practices needed to integrate CS into the math curriculum through educational robotics, we drew on two areas of research: research-based principles of teacher PD in technology integration (e.g., Kopcha, 2012; Mouza, 2009) and Smith & Stein's (2011) five practices for mathematical discourse.

Research-based Technology Integration PD Principles

Teachers learning to integrate technology face a variety of potential barriers, including a lack of knowledge, beliefs, time, and support (Kopcha, 2012). Mouza (2009) developed a set of research-based principles for technology PD that address those barriers. The six principles include: focus on teacher knowledge, reform-type activities (e.g., co-teaching and co-planning), situate activities in teacher needs, active learning, extensive duration, and collective participation (e.g., regular meetings; inclusion of STEM coaches and technology specialists). These principles can



(a) (b)

Figure 1. Ozobot programming with (a) solid lines and color codes and (b) block-based programming.

help teachers manage barriers such as access, knowledge, and skills, which, in turn, improves their technology use in the classroom (Kopcha, 2012; Mouza, 2009).

Our PD emphasized the six principles. It began with a week-long summer workshop in which teachers completed five activities *as if they were the student*. We chose this strategy to promote *active learning* as part of our PD. For example, the teachers used the robots to draw and name geometric shapes for the robot to follow, explore the relationship between perimeter and area, and program the robot to trace various angle measures, line lengths, and regular polygons. These activities were hands-on and provided multiple opportunities for the teachers to apply computer programming skills and mathematical thinking in a similar manner as their students. This strategy is similar to that used by Carpenter et al. (1989) who found that teachers who engaged in the same activities as their students during PD were better equipped to support and guide student learning in the classroom.

As part of the workshop, the teachers also developed three lesson plans that integrated educational robotics into the math curriculum; the goal was for the teachers to make a plan for using the educational robots over the coming academic year. This helped them *situate the PD in their own needs* and the needs of their students. *Extensive duration* was achieved through monthly follow-up meetings and in-classroom support throughout the year as the teachers implemented their lessons. *Collective participation* was established in that teachers received in-classroom support from both the researchers and the on-sight math coach and technology specialist. In-classroom support, which is a *reform-type activity*, primarily consisted of co-teaching. When co-teaching, the teacher led the lesson activity while in-the-moment coaching and modeling were provided by one of the researchers and/or the math coach or technology specialist. That coaching focused primarily on pedagogical approaches to supporting mathematical thinking through Smith and Stein's (2011) five practices for orchestrating mathematical discourse (see next section).

Five Practices for Mathematical Discourse

Throughout the PD, teacher learning centered on Smith and Stein's (2011) five practices for orchestrating mathematical discourse. The five practices encourage teachers to *anticipate* student misconceptions, *monitor* student activity for those misconceptions, and *select* and *sequence* student work in a way that supports *connecting* student thinking to larger mathematical concepts. These practices encourage teachers to help students solve mathematical problems without providing the information directly by testing conjectures mathematically, trying new problem-solving approaches and ideas, and struggling productively with math concepts.

We chose Smith and Stein's (2011) five practices because those practices support the teaching of math through educational robotics at the elementary level. For example, Ozobot block programming uses circular symbols to indicate angle measures, where a 90° turn is represented as a circle that is one quarter full. A teacher using the five practices would *anticipate* that young children might struggle with representing a 90° angle as one quarter of a circle and develop questions that encourage students to struggle with the concept (e.g., Which block code represents 90°? What portion of the circle is represented? How do you know?). The teacher could *select* and *sequence* examples of that concept from student work, building on the idea that two 90° turns add up to 180°. This *connects* with the concept of fractional addition as represented visually (e.g., two quarter circles is half a circle). In this way, the teacher supports students in applying mathematical concepts while engaging in CT skills like recognizing patterns and engaging in algorithmic thinking (e.g., 90° + 90° is half a circle or 180°, which can also be expressed as ¼ + ¼ = ½).

Research suggests that the five practices can help teachers sustain cognitive demand around mathematical concepts (Boston & Smith, 2009) and support students in deeper levels of mathematical reasoning (Ball et al., 2008; Lampert et al., 2010). We anticipated that blending Smith and Stein's practices with research-based principles of PD would help improve our teachers' attitudes and skills when integrating educational robotics into elementary math curriculum.

METHODS

This study took place in a high-need elementary STEM charter school in the rural South. Leadership identified improving students' mathematical skills through CS activity as an important goal. The school, which serves over 300 diverse K-5 students (51% African-American, 38% Caucasian, 6% Hispanic), partnered with the university to offer PD through a US Department of Education *Improving Teacher Quality* state grant.

Participants

The participants were 12 elementary teachers. Eight were grade-level teachers from first (1 teacher), second (1), third (2), fourth (2) and fifth (2) grade. The remaining participants included the school library media specialist, who co-taught with teachers each week in the media center, and three STEM coaches, who regularly provided individual, in-classroom support to the teachers. Aside from the school library media specialist, none of the teachers had prior experience with educational robotics.

Data Collection and Analysis

Teachers completed a survey and test of knowledge during the first professional development session as a pre-

measure, then again 10 months later as a post-measure. Lesson plans were analyzed at the conclusion of the PD. Each form of data collection is described below.

Survey

The survey included five attitudinal items about a teacher's confidence in integrating robotics into the math curriculum. Teachers rated their agreement on a five-point Likert scale from 1 (*Strongly Disagree*) to 5 (*Strongly Agree*). Wilcoxon Signed-Ranked tests were conducted on each item to evaluate changes from pre- to post-PD. In order to reduce Type I error, the significance level was set at .01 by using a Bonferroni correction and dividing .05 by the number of items (5).

Test of Knowledge

The test of knowledge addressed two areas: lesson planning and computer programming. Teachers first provided a short answer in which they applied Smith and Stein's (2011) five practices to a pre-made lesson scenario: "You plan to have students program a robot to trace a polygon that has three equal sides. To the best of your current knowledge and abilities, complete the elements of the Thinking Through a Lesson Plan (Smith & Stein, 2011) protocol that are listed below." Responses were scored from *low* (1) to *high* (3) evidence of knowledge associated with the five practices (anticipating, monitoring, selecting, sequencing, connecting). Scores were assigned based on the number of accurate possibilities offered (e.g., 2+ was high, 1 was mid, inaccurate or no response was low). For example, a high score in *connect* meant teachers noted at least two other related math concepts (e.g., the number of degrees in a circle; representing parts of a whole circle as a fraction). To test CS knowledge, teachers were provided with two examples of block-based computer code and asked to predict the results of that code. The two examples addressed the programming of basic movement skills to form a square shape and higher levels of logic that included loops and conditionals. Teacher responses were scored from high (3), or entirely correct, to low (1), or entirely incorrect.

Throughout the scoring of the test items, two independent raters coded the CS test items to establish inter-rater reliability, which was 90% and deemed satisfactory to proceed with the first rater's coding. Wilcoxon Signed-Ranked tests were conducted on each item to evaluate changes from pre- to post-PD. Because of the large number of items associated with lesson planning, we reduced Type I error by setting the significance level at .01 using a Bonferroni correction and dividing .05 by the number of items (5).

Lesson Plan Evaluation

Nine unique lessons were created by the teachers in this study. Each was evaluated using Schoenfeld et al.'s (2014) *Teaching for Robust Understanding of Mathematics Rubric (TRU)*. The TRU rubric assesses the strength of a lesson plan on a scale from *low* (1) to *high* (3) across five dimensions: mathematics, cognitive demand, student access to mathematics, student agency, and the assessment. Mathematics refers to the way that lesson activities support meaningful connections between concepts and procedures and offer opportunities to build a coherent understanding of math. Cognitive demand refers to the presence of the Smith and Stein teaching practices (e.g., anticipating, monitoring); this relates to student access in that classroom activity structures need to invite and support students' active engagement in core concepts and ideas. Agency refers to opportunities for students to conjecture, test, and modify ideas, whereas assessment refers to students having opportunity to explain their thinking and build off of initial ideas or address key misconceptions. Scores were applied initially by two independent raters who then compared ratings; scores were then averaged between the two reviewers so that disagreements between reviewers were factored into the final scores.

RESULTS

As shown in Table 1, scores on the five survey items were compared before and after the PD using a Wilcoxon Signed-Ranks test. Results indicated that teachers ranked their ability to use robots to teach math as being significantly higher after the workshop (M = 3.83; Mdn = 4) than before (*M* = 2.42; Mdn = 2), *W* = 0, $p < .01$. Teachers also ranked their ability to help students when having difficult with the robot significantly higher after the PD (M= 3.70; Mdn = 3) than before (M = 2.55; Mdn = 3), W = 1, $p < .01$. Teachers confidence in teaching math through robotics was significantly higher after the PD (*M* = 3.70; *Mdn* = 3) than before (*M* = 2.55; *Mdn* = 3), *W* = 4, $p < .01$. No other statistically significant results were found.

As shown in Table 2, the scores associated with each of Smith and Stein's (2011) practices for orchestrating productive mathematical discourse were significantly higher after the PD than before. This included their ability to *anticipate* student responses and challenges (*Mdn* = 2 and 0.75, *W* = 1.5), monitoring student work (*Mdn* = 2 and 1; *W* = 2), select and sequence student work for whole-class discussion (*Mdn* = 2 and 0; *W* = 2), and connecting student ideas with other mathematical concepts (*Mdn* = 2 and 0.75; *W* = 0). The difference in rankings for both CS items was not statistically significant.

As shown in Table 3, teacher lesson plans addressed a number of K-5 mathematical concepts (e.g., constructing fractions; calculating distance and perimeter; dividing shapes in half). The lesson plans approached a score of 'high' (3.00) with regard to their incorporation of mathematical concepts and ideas (*M* = 2.78), opportunities to conjecture and test or modify ideas (i.e., agency; *M* = 2.89), and assessing learning by building off initial ideas and/or misconceptions (*M* = 2.78).

Table 1. Mean and Median Scores by Survey Item

Item	Pretest		Posttest	
	Mean	Median	Mean	Median
I can consistently use robots to teach math.*	2.42	2	3.83	4
I can help students when they have difficulty with the robots.*	2.50	3	3.67	3.5
I have the skills necessary to use robotics to teach specific math concepts and skills.*	2.42	2	3.67	4
I can regularly incorporate robots into math when appropriate.	3.33	3	4.00	4
I am confident I can teach math concepts/skills with robotics.	4.00	4	4.08	4

* Statistically significant, $p < .01$.

Table 2. Mean and Median Scores by Test Item

Item	Pretest		Posttest	
	Mean	Median	Mean	Median
Lesson Planning: Anticipating*	0.58	0.75	1.96	2
Lesson Planning: Monitoring*	0.96	1	2.13	2
Lesson Planning: Selecting & Sequencing*	0.25	0	1.75	2
Lesson Planning: Connecting*	0.58	0.75	1.67	2
CS skill: Basic movement	2.21	2	2.54	2.50
CS skill: Loops and conditionals	1.21	1	1.27	1

*Statistically significant, *$p < .01$,

Table 3. Mean TRU Scores by Lesson Plan and Associated CS Activity and Math Concepts

Title	Overview	Mathematics	Grade	Math	Cog Dem.	Content	Agency	Assess
Fraction Bowling	Knock down pins and name score as a fraction. Graph / interpret.	Fractions	3-5	3	3	3	3	3
Shape of Things	Draw shapes on paper; use color-codes.	Shapes and angles	1-5	2	2	2	3	3
Addition	Move through an addition maze.	Add within 100	1	1	1	2	3	2
Fraction Shapes	Create a geometric shape and divide into halves and quarters	Express equal shares	1	3	3	3	3	3
Three Little Pigs Measuring	Create path and measure using a non-standard method.	Express length	1	3	2	3	3	3
Multiply Maze	Create a path with total product < 5,500.	Multiply multi-digit numbers	3	2	2	3	2	2
National Park Data	Merge two maps to plot course.	Read a key, interpret data	3	2	3	3	3	3
Santa's Dilemma	Plot course and program Ozobot to follow.	Multiplication and division	3, 5	3	3	3	3	3
Ozobot Race	Find the shortest path.	Factoring; perimeter and area	3, 5	3	3	3	3	3

For example, the lesson, Shape of Things, was created by the school library media specialist in collaboration with the first grade teacher, then shared with other grades as an introduction to visual programming. The overarching goal was to use the Ozobot to reinforce key geometry standards and concepts. Student agency was strong, as the students explored their own shapes and actively talked about the connections with geometry based on their own drawings.

Teachers from the other grade levels used the core lesson and modified it to meet the needs of their own students. For example, the teachers in grades K-2 supported agency by planning to ask questions about equal and unequal side lengths (i.e., regular vs. irregular polygons) and how that equality and inequality might change the Ozobot's movement. The teachers in grades 3-5 supported agency by planning to use open-ended questions to challenge students to draw regular five- and six-sided polygons (e.g., "Can you draw a shape with five equal sides? How do you know the sides are equal?"). This connected with the idea that polygons with more sides are comprised of multiple isosceles or equilateral triangles.

Discussion

Our PD adopted a unique strategy—it combined research-based principles of technology integration PD with the practices associated with teaching mathematics. The results suggest that this approach helped our teachers improved their confidence and pedagogical knowledge for using educational robotics to teach. Those improvements were reflected in their lesson plans, which exhibited high levels of mathematical content and opportunities to conjecture about mathematics. Our results support others who found that active learning and follow-up support can lead to changes in classroom practices and behaviors that support CS education (Clark & Hollingsworth, 2002; Goode & Margolis, 2011). This is an important step towards reducing a teacher's perception that CS activities like educational robotics are a burden and increasing the integration of CS into the elementary curriculum (Ketelhut et al., 2020).

Teachers in our study did not improve their CS knowledge at a significant level. One reason may be that the teachers in our study preferred to focus on visual coding techniques with their students rather than a block-based programming environment. This would make sense in lower grade levels, where it would be easier for a teacher to have young children draw and color to program the robot than use a computer interface to create and download blocks of code. This is consistent with Zhang and Nouri (2019) who found that advanced programming skills are challenging for teachers to incorporate in their instruction and often overlooked because students struggle with them. As such, it is not surprising that CS knowledge did not see any significant gains in this study.

Implications

The improvements in teacher attitudes and knowledge noted in this study suggest how sustained PD with reform-type activities (e.g., monthly follow-up meetings; in-classroom support) can help teachers develop new skills and practices. In the week-long summer workshop, teachers engaged in *active learning*, programming the robots themselves to construct geometric shapes, explore the relationship between area and perimeter, and associate distance with the measurement of length. They then planned lessons for their own students (*situated in their needs*). Over the academic year, they met with the research team regularly (*extended duration*) and received in-classroom support through co-teaching (*reform-type activities*). These activities are likely to have contributed to the teachers' gains in confidence and knowledge of teaching practices associated with educational robotics in this study. Similar to the current study, Mouza (2009) and Kopcha (2012) both found that ongoing, in-classroom support led to improved attitudes toward and practices with technology. Likewise, Menekse (2015), reported that PD in the context of CS should be sustained over time and place emphasis on active learning and teacher's knowledge of effective teaching strategies.

Conclusion

This paper offers a detailed account of teacher PD in CS education that can support teacher change. While the sample size is small, it is likely that the approach employed in this study can be transferred to a new context to foster teacher attitudes and lesson planning skills when using educational robots to teach both CS and mathematics. Future research on teacher PD would benefit from adding teacher observations to see how mathematics is supported during educational robotics. Doing so is an important step towards increasing participation in CS and making CS accessible for all.

Acknowledgments

Funding: This study was supported by the US Department of Education's *Improving Teacher Quality State Grant* program.

Keywords: elementary teacher professional development, teacher technology use, computer science education, educational robotics in mathematics

Theodore J. Kopcha
The University of Georgia
205 River's Crossing
850 College Station Road
Athens, GA 30602
Email: *tjkopcha@uga.edu*

References

Anwar, S., Bascou, N. A., Menekse, M., & Kardgar, A. (2019). A Systematic Review of Studies on Educational Robotics. *Journal of Pre-College Engineering Education Research (J-PEER)*, 9(2), Article 2. https://doi.org/10.7771/2157-9288.1223

Ball, D. L., Thames, M. H., & Phelps, G. (2008). Content knowledge for teaching: what makes it special? *Journal of Teacher Education, 59*(5), 389-407. https://doi.org/10.1177/0022487108324554

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

Boston, M. D., & Smith, M. S. (2009). Transforming secondary mathematics teaching: Increasing the cognitive demands of instructional tasks used in teachers' classrooms. *Journal for Research in Mathematics Education, 40*(2), 119-156. https://doi.org/10.2307/40539329

Carpenter, T. P., Fennema, E., Peterson, P. L., Chiang, C. P., & Loef, M. (1989). Using knowledge of children's mathematics thinking in classroom teaching: An experimental study. *American Educational Research Journal, 26*(4), 499-531.

Dunbar, K. M., & Rich, K. M. (2020). Mathematics Makes Robots Roll. *Mathematics Teacher: Learning and Teaching PK-12, 113*(7), 565-572

Eguchi, A. (2017). Bringing robotics in classrooms. In *Robotics in STEM education* (pp. 3-31). Springer, Cham.

Grover, S., & Pea, R. (2013). Computational thinking in K–12: A review of the state of the field. *Educational researcher, 42*(1), 38-43. https://doi.org/10.3102/0013189X12463051

Karp, T., & Maloney, P. (2013). Exciting Young Students in Grades K-8 about STEM through an Afterschool Robotics Challenge. *American Journal of Engineering Education, 4*(1), 39-54.

Ketelhut, D. J., Mills, K., Hestness, E., Cabrera, L., Plane, J., & McGinnis, J. R. (2020). Teacher change following a professional development experience in integrating computational thinking into elementary science. *Journal of science education and technology, 29*(1), 174-188.

Kopcha, T. J. (2012). Teachers' perceptions of the barriers to technology integration and practices with technology under situated professional development. *Computers & Education, 59*(4), 1109-1121.

Lampert, M., Beasley, H., Ghousseini, H., Kazemi, E., & Franke, M. (2010). Using designed instructional activities to enable novices to manage ambitious mathematics teaching. In *Instructional explanations in the disciplines* (pp. 129-141). Springer US.

Lee, I. A., Psaila Dombrowski, M., & Angel, E. (2017). Preparing stem teachers to offer New Mexico computer science for all. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education* (pp. 363-368).

Leonard, J., Mitchell, M., Barnes-Johnson, J., Unertl, A., Outka-Hill, J., Robinson, R., & Hester-Croff, C. (2018). Preparing teachers to engage rural students in computational thinking through robotics, game design, and culturally responsive teaching. *Journal of Teacher Education, 69*(4), 386-407. https://doi.org/10.1177/0022487117732317

Menekse, M. (2015). Computer science teacher professional development in the United States: a review of studies published between 2004 and 2014, *Computer Science Education, 25*:4, 325-350. http://dx.doi.org/10.1080/08993408.2015.1111645

Mouza, C. (2009). Does research-based professional development make a difference? A longitudinal investigation of teacher learning in technology integration. *Teachers College Record, 111*(5), 1195-1241.

National Science & Technology Council (2018). Charting a course for success: America's strategy for STEM education. Retrieved https://www.whitehouse.gov/wp-content/uploads/2018/12/STEM-Education-Strategic-Plan-2018.pdf

Papert, S. (1980). *Mindstorms*. New York: Basic Rooks.

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. Waynesville, NC USA: Society for Information Technology & Teacher Education. Retrieved March 12, 2021 from https://www.learntechlib.org/primary/p/207487/

Schoenfeld, A. H., Floden, R. E., & the Algebra Teaching Study and Mathematics Assessment Project. (2014). An introduction to the TRU Math document suite. Berkeley, CA & E. Lansing, MI: Graduate. School of Education, University of California, Berkeley & College of Education, Michigan State. University. Retrieved: http://map.mathshell.org/trumath.php

Smith, M., & Stein, M. K. (2011). Five practices for orchestrating productive mathematical discourse. *Reston, VA: National Council of Teachers of Mathematics*.

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

Wing, J. (2011). Research notebook: Computational thinking—What and why. *The link magazine, 6*.

Yadav, A., Gretter, S., Good, J., & McLean, T. (2017). Computational thinking in teacher education. In *Emerging research, practice, and policy on computational thinking* (pp. 205-220). Springer, Cham.

Yuan, J., Kim, C., Hill, R., & Kim, D. (2019). Robotics integration for learning with technology. *Contemporary Issues in Technology and Teacher Education 19*(4), 708-735.

Yuen, T., Stone, J., Davis, D., Gomez, A., Guillen, A., Tiger, E. P., & Boecking, M. (2015). A model of how children construct knowledge and understanding of engineering design within robotics focused contexts. *International Journal of Research Studies in Educational Technology, 5*(1).

Zhang, L., & Nouri, J. (2019). A systematic review of learning computational thinking through Scratch in K-9. *Computers & Education, 141*, 103607. https://doi.org/10.1016/j.compedu.2019.103607

Integration of Computational Thinking Into English Language Arts

Sharin Rawhiya Jacob, Miranda C. Parker, and Mark Warschauer, *University of California, Irvine*

Corresponding Author: Sharin Rawhiya Jacob, *Sharinj@uci.edu*

Abstract

This paper describes the development and implementation of a yearlong integrated English Language Arts (ELA) and computational thinking (CT) curriculum that has been adapted to meet the needs of multilingual students. The integration of computational thinking into K-12 literacy instruction has only been examined in a handful of studies, and little is known about how such integration supports the development of CT for multilingual students. We conducted a qualitative case study on curricular implementation in a general education classroom with large numbers of students designated as English learners. Results from detailed field notes revealed that the strategic application of instructional practices was implemented in the service of building on students' existing literacy skills to teach CT concepts and dispositions. The CT and literacy framework put forth in this study can be used as an analytic framework to highlight how instructional strategies mobilize the existing literacy and CT resources of linguistically diverse students. Based on our findings, we discuss recommendations for future integrated ELA-CT curricula.

While the integration of computational thinking (CT) into science, technology, engineering, and mathematics (STEM) education has been well studied (Jona et al., 2014; Sengupta et al., 2018; Weintrop et al., 2016), there is a smaller but growing body of work on CT and literacy integration (Jacob et al., 2018; Burke & Kafai, 2012; Kafai et al., 2020; Vogel et al., 2020). There are several affordances to engaging diverse learners when combining CT and literacy instruction. Programming in narrative genres may foster literacy development and technological fluency while motivating students who may not otherwise identify with computer science (CS; Burke & Kafai, 2012). This can facilitate the kinds of inquiry, cultural and community engagement, and social recognition that are integral to fostering identity development in STEM (National Research Council [NRC], 2014).

Computational thinking and literacy integration is particularly beneficial in elementary grades, as instructional minutes allotted to STEM are extremely limited, especially for students who are second language learners (Dorph et al., 2011). While the value of focusing on language and literacy instruction in early grades is undisputed, integration of CT within the language arts curriculum can provide a way to overcome STEM instructional time constraints, allowing students to get vital early exposure to CS while also supporting their language development.

This paper describes the implementation of an English Language Arts (ELA)-focused curriculum to support learning and positive identification with CS among multilingual elementary school students. We first describe the model of computational literacies we draw on and then describe the curriculum that forms the basis of the intervention and study.

We address the following research question:

A. What strategies are used by upper elementary teachers to integrate CT into literacy and language instruction?
B. How does applying the CT and literacy framework advance our understanding of how to leverage multilingual students' literacy resources to develop their computational thinking skills?

Computational Literacies

Our study draws from Jacob and Warschauer's (2018) model of computational literacy, which situates computational thinking as a fundamental literacy required for full societal participation (cf. diSessa, 2000; Wing, 2006). This model proposes three dimensions for 1) characterizing the relationship between computational thinking and literacy (i.e., computational thinking *as* literacy), 2) examining how students' existing literacy skills can be leveraged to foster computational thinking (i.e., computational thinking *through* literacy), and 3) discussing the ways in which computational thinking skills foster literacy development (i.e., literacy *through* computational thinking; Jacob & Warschauer, 2018; see Figure 1).

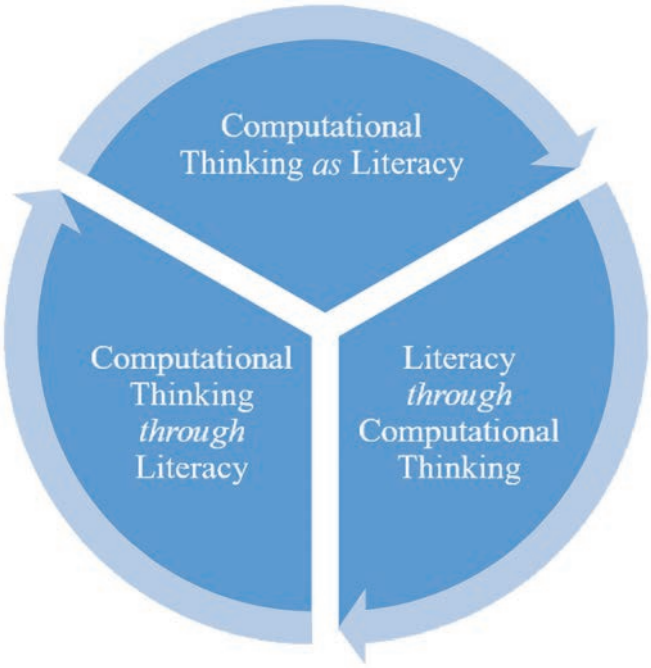


Figure 1. A Three-Dimensional Framework for Understanding Computational Thinking and Literacy (Jacob & Warschauer, 2018)

For the purpose of this paper, we focus on the second component of the computational thinking and literacy framework: computational thinking *through* literacy. To this end, we examine how students leverage their existing literacy skills as a mechanism for learning computational thinking. Integrating computational thinking into ELA content has multiple affordances for CT learning. Evidence suggests that learning to read and write and to code can go hand in hand (Peppler & Warschauer, 2011; Bers, 2019). The several interlocking features of coding and literacy draw children's attention to symbol-meaning relationships. For example, students interact with text in multiple ways as they use Scratch and leverage their knowledge of multimodal signifiers to assemble programs. These relationships offer a highly engaging and supportive environment for children with emerging literacies to demonstrate their skills and abilities (Peppler & Warschauer, 2011).

Additionally, informational and narrative genres capture the semiotic process related to computing. To illustrate, Burke and Kafai (2012) leveraged students' knowledge of the writing process (i.e., drafting, revising, editing) to engage them in designing computational artifacts (i.e., (design, troubleshooting, debugging). Similarly, De Souza et al. (2011) compared students' narrative accounts of programming games to their design process, paying specific attention to verbal structures. Findings indicated that at first students used transitive verb based narrative accounts to design games, and over time they began to use intransitive verbal structures that more closely resembled programming languages For example, A typical student characterization of a game "the hunter killed the monkey" was actually programmed as "the monkey disappears when it touches the hunter." Results such as these suggest that students' existing literacy skills can be mobilized to develop their computational thinking skills.

The CS-ELA Integrated CT Curriculum

Elementary schools with large percentages of multilingual students, not surprisingly, devote large amounts of instructional time to improving students' English skills. This makes it challenging to introduce non-core curriculum, such as CS. Indeed, research has shown that science, let alone CS, is rare in high-ELL schools and districts (Gomez-Zwiep, 2017). Our project has addressed this challenge by adapting the Creative Computing Curriculum (Brennan et al., 2014) for integration into ELA instruction. The curriculum--called Elementary Computing for All--exploits the affordances of Scratch for learning to decode and code stories of the same genres that are emphasized in traditional narrative and informative texts in elementary school. It also integrates age-appropriate readings about diverse pioneers in CS, thus strengthening the connection to reading while also providing culturally relevant support. In this way, STEM identity is developed as children learn about diverse computer scientists and code stories about their own lives and communities.

The storybooks integrated into the curriculum teach not only computational thinking concepts but also key dispositions that foster student success in computing. In 2011, the International Society for Technology in Education (ISTE) and the Computer Science Teachers Association (CSTA) outlined specific dispositions or mindsets that are fundamental to student success in computational thinking including 1) confidence in dealing with complexity, 2) persistence in working with difficult problems, 3) tolerance for ambiguity, and 4) the ability to deal with open ended problems (ISTE & CSTA, 2011). The storybooks in our curriculum teach these dispositions in culturally and age appropriate ways. For example, students read *The Most Magnificent Thing*, a storybook about a young girl who, through engaging in making activities, acquires positive dispositions and approaches to computing. The protagonist of the book desires to construct a computational artifact for her dog. Throughout the design process, she abstracts her model, decomposes her problem, implements her solutions, debugs her errors, and engages in iterative problem solving to arrive at a "magnificent" solution. To this end, the storybook teaches both computational thinking concepts such as abstraction, iteration, decomposition, and debugging as well as dispositions that enable students to become successful computational thinkers. The big idea of the story, having a growth mindset, is operationalized through examples of the protagonist dealing with complex problems, persisting through mistakes, and tolerating ambiguity. Storybooks such as these provide affordances

for teaching both the computing concepts necessary for learning the discipline as well and dispositions that foster successful computational thinkers.

Linguistic Scaffolding

Researchers and practitioners worked collaboratively to develop additional language scaffolding to amplify the curriculum's effectiveness with multilingual students, following effective practices recommended by a national panel (National Academies of Science, Engineering, and Medicine [NASEM], 2018). First, the revised curriculum integrates CS and ELA tasks to **engage students in disciplinary practices**. Students explore and modify existing programs before creating their own projects. These kinds of structured inquiry-based science approaches provide a powerful mechanism for providing authentic contexts for language use (NRC, 1996) while making instruction more engaging, concrete, and meaningful for multilingual students (Janzen, 2008; NRC, 2012; Rosebery & Warren, 2008). Computer science disciplinary activities and learning goals are aligned with standards to guide teachers (see Table 1 for an example).

Second, the revised curriculum **encourages rich classroom discourse** through explicit suggestions of collaborative activity formats to invite students to use their everyday sense-making and disciplinary language in multiple contexts (Shea & Shanahan, 2011).

Third, strategies that teachers can use to **build on students' existing resources** (i.e., cultural, linguistic, semiotic, embodied) to acquire proficiency in language and CS are explicitly stated in the curriculum and during professional development. For example, the curriculum

Table 1. Sample Learning Goals That Integrate Grade 4 Common Core ELA, English Language Development, and Computer Science Teachers Association Standards

Activity: Students program a story about their lives, families, or communities		
Computer Science Concepts: Loops, Sequences, Conditionals		
Computer Science Teachers Association (CSTA) Standards		
CSTA 1B-AP-10	Create programs that include sequences, events, loops, and conditionals	
CSTA 1B-AP-13	Use an iterative process to plan the development of a program by including others' perspectives and considering user preferences	
CSTA 1B-AP-15	Test and debug a program or algorithm to ensure it runs as intended	
English Language Development (ELD) Standards		
Emerging	**Expanding**	**Bridging**
3. Offering opinions Negotiate with or persuade others in conversations using basic learned phrases (e.g., I think) as well as open responses in order to gain and/or hold the floor.	**3. Offering opinions** Negotiate with or persuade others in conversations using a variety of learned phrases (e.g., That's a good idea. However…) as well as open responses, in order to gain and/or hold the floor.	**3. Offering opinions** Negotiate with or persuade others in conversations using a variety of learned phrases (e.g., That's a good idea. However…) as well as open responses in order to gain and/or hold the floor, elaborate on an idea, and provide different opinions.
11. Supporting opinions Offer opinions and provide good reasons (e.g., My favorite book is X because X) referring to the text or to relevant background knowledge.	**11. Supporting opinions** Offer opinions and provide good reasons and some textual evidence or relevant background knowledge (e.g., paraphrased examples from text or knowledge of content).	**11. Supporting opinions** Offer opinions and provide good reasons with detailed textual evidence or relevant background knowledge (e.g., specific examples from text or knowledge of content).
Corresponding English Language Arts Standards		
CCSS.ELA-L.SL.4.1 **CCSS.ELA-L.SL.4.4** **CCSS.ELA-L.SL.4.6** **CCSS.ELA-L.W.4.9**	**Engage** effectively in a range of collaborative discussions with diverse partners, building on others' ideas and expressing their own clearly. **Report** on a topic or text, tell a story, or recount an experience in an organized manner, using appropriate facts and relevant, descriptive details to support main ideas or themes; speak clearly at an understandable pace. **Differentiate** between contexts that call for formal English (e.g., presenting ideas) and situations where informal discourse is appropriate (e.g., small-group discussion); use formal English when appropriate to task and situation. **Draw** evidence from literary or informational texts to support analysis, reflection, and research.	

and professional development include tips for teacher "talk moves'' (Michaels & O'Connor, 2015), namely asking for clarification and leveraging students' own ways of explaining to guide them towards more formal language and advanced CS concepts.

Fourth, visualizations and physical, unplugged activities are built into the curriculum to **engage students in multiple modalities**, including linguistic modalities of talk and text, as well as nonlinguistic modalities such as gestures, pictures, and symbols, to better teach key academic vocabulary and CT concepts (cf. Lee et al., 2019).

Fifth, the curriculum **provides explicit focus on how language functions in the discipline** by providing language frames to teachers for use by students during peer feedback and pair programming, and while asking for assistance (see example in Table 2).

METHODS

Researchers at Western University (pseudonym) and educators in a large urban school district joined together in a research-practice partnership to iteratively develop and implement the curriculum. The district has among the highest percentages in the nation of Latinx students (93%), low-income learners (89.7% receiving free or reduced-price lunch), and students designated as English language learners (62.7% in the elementary grades). Ordinary elementary school teachers in the district taught the curriculum in their own classes after a one-week professional development program in the summer that taught them about Scratch, computational thinking, equity issues in CS education, and the CT-ELA approach.

Though broader data were collected for the larger study, in this paper we only focus on the instructional strategies carried out by teachers to integrate CT and ELA instruction that meets the needs of multilingual students. Thus, we analyze structured notes from weekly classroom observations. For each field observation, two researchers took detailed field notes on teachers' instructional moves, students' interaction, and computing tasks and activities. Four Ph.D. students and three undergraduates observed teachers' classes when they integrated CT and literacy lessons. All lessons were audio recorded and transcribed.

These data were analyzed through open coding in iterative cycles. Two researchers collaborated to assign initial codes to excerpts of text that pertained to strategies used by teachers to integrate CT and ELA content (Hsieh & Shannon, 2005), paying specific attention to instructional practices that are effective for engaging multilingual students in STEM (NASEM, 2018). After coding 25% of field notes, the researchers met to combine, split, and categorize codes based on initial findings. After this discussion, the first author applied the consolidated codes to the rest of the data, generating new codes when they were pertinent to the research questions. After coding all of the field notes, two researchers (first and second author) randomly selected 10% of the data to conduct an interrater reliability check and achieved 83% agreement. The two researchers then met to discuss differing codes and redefine each of the codes. After revising the codebook, they reapplied the modified codes and reached 94% agreement.

RESULTS

All the teachers in our study were able to successfully teach the curriculum and carry out appropriate strategies for students designated as English learners that integrated CT and ELA in the classroom (see Table 3). To illustrate this, we present a case from one classroom taught by Jenny (pseudonym), which was a general education classroom of predominantly Latinx and low-income students designated as English learners.

Table 2. Computer Science Language Functions

Teacher Activities	Student Discourse			CS Concepts (Language Function)
	Emerging	Expanding	Bridging	
Remind students to think about the events that will cause each action to happen in their project, which programs will run parallel to each other, and how their project will reset once it has finished running.	I need help with __. __ caused __ to happen. __ and__ are running at the same time. I used __ to reset the program.	I am having difficulty with __. __ is the event that caused __ to happen. __ and __ are running parallel to each other. I used __ to initialize the program.	Could you help me fix the following challenge in my code __? The event that caused __ to happen is __. __ and __ are running parallel to each other/ simultaneously/at the same time. __ caused the program to initialize.	Debugging, events, initialization, parallelism (Describing, comparing)

Table 3. Coding Framework Excerpt

Categories	Strategies for Activating Prior Knowledge	Strategies for Asking Questions	Strategies for Providing Direct Instruction	Strategies for Providing Language Support
Sample Codes and Definitions	Leveraging Students' Background Knowledge *Applying students' existing knowledge to lesson content.* Building on Students' Personal Experiences *Connecting lessons to students' personal experiences.*	Using Questions to Foster Higher Order Thinking *Asking higher order questions (i.e., analysis, evaluation) instead of recall of comprehension checks.* Using Questions to Make Interdisciplinary Connections *Asking how one subject is similar to another (e.g., using elements of storytelling to describe coding processes).* Using Questions to Elicit Big Idea *Asking how instructional materials relate to the big idea of the lesson.*	Discussing Computational Concepts *Discussing computational concepts (i.e., abstraction, algorithms) and programming concepts (i.e., sequence, loops, conditionals).* Pre-Teaching Lesson Vocabulary *Introducing lesson vocabulary in multiple modalities at beginning of lesson.*	Facilitating Discourse Through Collaboration *Engaging students in peer-to-peer or teacher-student-student talk to build on students' existing resources.* Prompting Students to Use Sentence Frames *Using sentence frames as prompts to provide language support, guidance, and to encourage elaboration.* Encourage Students to Use CS Language During Reflection *Encouraging students to use CS language gradually on their own.*

Strategies Used for CT-ELA Integration

Jenny's most frequently used strategy included multiple questioning techniques, and she made a point to integrate ELA reading strategies with CT lessons. For example, after reading *The Most Magnificent Thing* to her students, she used questioning techniques to check students' understanding of key computational thinking concepts such as sequencing, decomposition, debugging, and abstraction. She also used questions to elicit big ideas, such as developing a growth mindset. To illustrate, after the protagonist of the story *The Most Magnificent Thing* finished designing her computational artifact, the teacher asked: *"Was it perfect?"* The students responded: "No!" Then the teacher asked, *"But did it do the job?"* and the whole group responded *"Yes!"* In this example, she underscored for her students the idea that while they can always improve their work, they should also be proud of the artifacts that they have created. Research corroborates the idea that the design process is iterative and emphasis should be placed on process over product when developing computational artifacts (Ryoo et al., 2015). Finally, Jenny's use of multiple questioning techniques facilitated comprehension of CT and literacy content by providing opportunities for students to experience ideas in multiple ways. She primarily questioned students during whole group activities and used specific techniques related to ELA instruction such as encouraging higher order thinking (i.e., providing supporting evidence), elaborating components of storytelling (i.e., students identify plot, characters, setting, conflict, resolution), and invoking the big idea (i.e., identifying the main idea of the lesson).

Jenny also facilitated student discourse by engaging them in collaboration during pair programming activities. For example, she instructed her students to provide constructive feedback to their peers, even if their peers' projects contained mistakes. This helped to normalize the making of mistakes in the classroom and foster persistence in the face of challenges. Another strategy Jenny used was the activation of prior knowledge, which involves priming students' existing knowledge and providing prerequisite knowledge for students to understand lesson concepts. To this end, Jenny would reference previous CT lessons to connect to the current lesson she was teaching. This strategy is essential to providing a foundation for multilingual students to assimilate new information (Lee & Fradd, 1998; Turner & Bustillos, 2017). Jenny also promoted the use of discipline-specific discourse by fostering interaction, prompting student reflection during whole group discussion, and modeling the use of CS language during whole group instruction.

Applying a CT and Literacy Framework Through CT-ELA Integration

We present a vignette that explores how the instructional moves employed by Jenny apply the CT and literacy framework (Jacob & Warschauer, 2018) to integrate CT-ELA

instruction for multilingual students. The purpose of this section is to advance our understanding of how teacher moves can benefit culturally and linguistically diverse students in a CT-ELA integrated curriculum.

Teaching CT and Literacy in Jenny's Diverse General Education Classroom

In the excerpt below (Table 4), Jenny reads *The Most Magnificent Thing* to her students and pauses the story multiple times to question her students to emphasize the key idea.

In this excerpt, Jenny is teaching *computational thinking through literacy* by leveraging students' knowledge of storytelling and narrative devices to engage them in productive discussion of computing concepts and dispositions. Using well-established techniques such as making predictions and discussing main ideas (Wright & Gotwals, 2018) allows Jenny's students to check her students' understanding of CT and literary concepts, through whole group interaction that is broken down into meaningful chunks. Through her questions, Jenny encourages students to engage in several CT concepts and practices, including sequences ("What happened next?"), abstraction ("What did she notice about all of those things?"), and experimenting and iterating ("Was it perfect?"). With this process she simultaneously teaches literary themes (i.e., plot, character development, conflict, resolution, theme), CT concepts (i.e., iteration, testing, debugging, design process), and positive attitudes and dispositions towards CT (i.e., growth mindset, confidence, perseverance). In her next lesson, Jenny moves on to apply the idea of a growth mindset to students' programming tasks, encouraging students to iterate and debug

Table 4. Audio Transcript of Jenny Teaching The Most Magnificent Thing

Speaker	Audio Transcript
Jenny:	(teacher pauses story) Why is she quitting? Talk to your partner. Why is she quitting? Tell me, why is she giving up? (students are busy discussing with one another)
Student 1:	It is too hard…
Student 2:	Not the way she wants it to be…
Student 3:	Maybe because what she is thinking it is not possible because it is hard (teacher resumes story then teacher pauses story again)
Jenny:	So tell me first of all, what was the problem with what she was building? What was she building?
Student 4:	A robot…
Student 5:	A car…
Jenny:	(Jenny plays the story to find out what she is building) What did she do? What happened first? What did she do first?
Student 4:	She got mad.
Jenny:	What happened next? Did she just stay mad and give up? What happened next?
Student 3:	She took her dog out for a walk and saw all that she did and what she gave up.
Student 6	So she looked at all of her work that she thought was wrong.
Jenny:	And what did she notice about all of those things?
Student 3:	There were pieces that she liked.
Student 7:	There were the right pieces that she made.
Jenny:	So she had to do what? To her thinking? She had to do what to her thinking?
Student 8:	She had to look at her invention.
Student 9:	Think more…
Student 10:	Think about her problems so that she could fix them…
Student 3:	Rethink her model…
Jenny:	And what happened at the end?...
Student 8:	She found out that she used different things but then she went back to change it and made it right.
Jenny:	Think about that last page. Was it perfect?
Whole Class:	No!!
Jenny:	But did it do the job?
Whole Class:	Yes!!!

challenging problems. In doing so, she phrased different questions to prompt students to think about examples and non-examples of growth mindset to provide students a framework for giving constructive feedback. Jenny's questioning techniques built on students' resources to make connections between pre-existing knowledge and new lesson content. By leveraging their existing resources, she assigns value to students' experiences and draws upon their funds of knowledge (Gonzalez et al., 2006).

Discussion

Our findings on instructional practices from Jenny's classroom can be used to support and inform strategies for the integrating CT, language, and literacy instruction. While there is a growing body of work on CT-ELA integrated curricula (Bers, 2019; Burke & Kafai, 2012), little research focuses on the instructional strategies that meet the specific needs of multilingual students (see Jacob et al., 2020). Typically, what has been missing in the literature is the specification of how the heterogeneous backgrounds of multilingual students influence their learning processes. Given that students in Jenny's class come from mostly Spanish speaking families and communities, but are schooled in English, their home language proficiency levels vary. Students from heterogeneous communities such as these tend to display proficiency in oral and written genres of informal English and leverage their everyday sense-making abilities to understand complex computational concepts (NASEM, 2018). To this end, Jenny's use of verbal questioning techniques to scaffold the children's storybook enabled her students to mobilize their oral and semiotic resources to make sense of CS lesson concepts and content.

The strategic application of instructional practices was implemented in the service of building on students' existing literacy skills to teach CT (Jacob & Warschauer, 2018). This investigation stands in contrast to empirical studies focusing on how to integrate CT and ELA instruction. Emerging CT and literacy frameworks advance this discussion to situate computational thinking as a literacy in itself across multiple dimensions. However, what has been lacking is theoretical frameworks focusing on the overlap between CT, language, and literacy learning that informs instructional practices for culturally and linguistically diverse learners. The CT and literacy framework put forth by Jacob and Warschauer (2018) can be used as an analytic framework to highlight how instructional strategies mobilize the existing literacy and CT resources of students with heterogeneous linguistic needs.

Implications

Based on our findings, we suggest that practitioners apply strategies for teaching CT-ELA integrated instruction that leverages students' existing resources to foster CT, language, and literacy skills. Practitioners who integrate CT curricula with narrative genres can use students' knowledge of storytelling devices to teach CT concepts. When serving multilingual students, teachers should also be aware of students' heterogeneous backgrounds. For students who are learning English and their home language at the same time, instruction that leverages their everyday language solidifies CS knowledge in preparation for engaging students in more demanding scientific and technical language. Finally, CS content should not be taught to the exclusion of the dispositions that will enable students to develop a sense of efficacy and belonging as computer scientists. Therefore, supplementing the curriculum with instructional materials, such as children's books, about diverse pioneers in the field of CS who persevere in the face of adversity is an excellent way to foster student identification with the discipline.

Acknowledgements

We would first like to thank the teachers and students who invited us into their classrooms and supported this work. We would also like to thank the reviewers for giving this work a platform. Finally, we would like to thank the National Science Foundation (grant 1738825 and 1923136) for providing the funding that made this project possible. Findings expressed in this work are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

Bers, M. U. (2019). Coding as another language: A pedagogical approach for teaching computer science in early childhood. *Journal of Computers in Education, 6*(4), 499-528.

Brennan, K., Balch, C., & Chung, M. (2014). Creative computing. Harvard Graduate School of Education. http://creativecomputing.gse.harvard.edu/guide/

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).

Connolly, J. H. (2001, July). *Context in the study of human languages and computer programming languages: A comparison*. In International and Interdisciplinary Conference on Modeling and Using Context (pp. 116-128). Springer.

diSessa, A. (2000). *Changing minds: Computers, learning and literacy*. MIT Press.

Dorph, R., Shields, P., Tiffany-Morales, J., Hartry, A., & McCaffrey, T. (2011). *High hopes–few opportunities: The status of elementary science education in California*. Sacramento, CA: The Center for the Future of Teaching and Learning at WestEd.

García, O., & Wei, L. (2015). Translanguaging, bilingualism, and bilingual education. *The Handbook of Bilingual and Multilingual Education*, 223, 240.

Gomez-Zwiep, S. (2017, March). *Creating equitable STEM spaces for English learners* [Paper presentation]. California STEM Includes Conference, Anaheim, CA, United States.

González, N., Moll, L. C., & Amanti, C. (Eds.). (2006). *Funds of knowledge: Theorizing practices in households, communities, and classrooms*. Routledge.

Hsieh, H. F., & Shannon, S. E. (2005). Three approaches to qualitative content analysis. *Qualitative Health Research*, 15(9), 1277-1288.

International Society for Technology in Education (ISTE) & Computer Science Teachers Association (CSTA), (2011). Operational definition of computational thinking for K-12 education.

Jacob, S. R., & Warschauer, M. (2018). Computational thinking and literacy. *Journal of Computer Science Integration, 1*(1), 1-19.

Jacob, S., Nguyen, H., Tofel-Grehl, C., Richardson, D., & Warschauer, M. (2018). Teaching computational thinking to English learners. *NYS TESOL journal, 5*(2), 12-24.

Jacob, S., Nguyen, H., Garcia, L., Richardson, D., & Warschauer, M. (2020, March). Teaching Computational Thinking to Multilingual Students through Inquiry-based Learning. In 2020 *Research on Equity and Sustained Participation in Engineering, Computing, and Technology* (RESPECT) (Vol. 1, pp. 1-8). IEEE.

Janzen, J. (2008). Teaching English language learners in the content areas. *Review of Educational Research, 78*(4), 1010-1038.

Jona, K., Wilensky, U., Trouille, L., Horn, M. S., Orton, K., Weintrop, D., & Beheshti, E. (2014, January). Embedding computational thinking in science, technology, engineering, and math (CT-STEM). In *Future directions in computer science education summit meeting, Orlando, FL*.

Kafai, Y., Proctor, C., & Lui, D. (2020). From theory bias to theory dialogue: Embracing cognitive, situated, and critical framings of computational thinking in K-12 CS education. *ACM Inroads, 11*(1), 44-53.

Lee, O., & Fradd, S. H. (1998). Science for all, including students from non-English-language backgrounds. *Educational Researcher, 27*(4), 12–21.

Lee, O., Llosa, L., Grapin, S., Haas, A., & Goggins, M. (2019). Science and language integration with English learners: A conceptual framework guiding instructional materials development. *Science Education, 103*(2), 317-337.

Michaels, S., & O'Connor, C. (2015). Conceptualizing talk moves as tools: Professional development approaches for academically productive discussions. In L. B. Resnick, C. Asterhan, & S. Clarke (Eds.), *Socializing intelligence through academic talk and dialogue* (pp. 347-362).

National Academies of Sciences, Engineering, and Medicine. (2018). *English learners in STEM subjects: Transforming classrooms, schools, and lives*. The National Academies Press. doi:h10.17226/25182

National Research Council. (1996). National science education standards. *ERIC Document Reproduction Service, 391*(690). National Academy Press.

National Research Council. (2012). *A framework for K–12 science education: Practices, crosscutting concepts, and core ideas*. National Academies Press.

National Research Council. (2014). *STEM integration in K-12 education: Status, prospects, and an agenda for research*. The National Academies Press. doi:10.17226/18612

Pane, J. F., & Myers, B. A. (2001). Studying the language and structure in non-programmers' solutions to programming problems. *International Journal of Human-Computer Studies, 54*(2), 237-264.

Peppler, K. A., & Warschauer, M. (2011). Uncovering literacies, disrupting stereotypes: Examining the (dis) abilities of a child learning to computer program and read. *International Journal of Learning and Media, 3*(3), 15-41.

Proctor, C. P., Dalton, B., & Grisham, D. L. (2007). Scaffolding English language learners and struggling readers in a universal literacy environment with embedded strategy instruction and vocabulary support. *Journal of Literacy Research, 39*(1), 71-93.

Rosebery, A. S., & Warren, B. (Eds.). (2008). *Teaching science to English language learners: Building on students' strengths*. National Science Teachers Association.

Ryoo, J. J., Bulalacao, N., Kekelis, L., McLeod, E., & Henriquez, B. (2015, September). Tinkering with "failure": Equity, learning, and the iterative design process. In *FabLearn 2015 Conference at Stanford University, September 2015*.

Sengupta, P., Dickes, A., & Farris, A. (2018). Toward a phenomenology of computational thinking in STEM education. *Computational Thinking in the STEM Disciplines*, 49-72.

Shea, L. M., & Shanahan, T. B. (2011). Methods and strategies: Talk strategies. *Science and Children, 49*(3), 62-66.

Turner, E. E., & Bustillos, L. M. (2017). Qué observamos aquí? Qué preguntas tienen? Problem solving in Ms. Bustillos's second-grade bilingual classroom. In *Access and equity: Promoting high quality mathematics in PK–2* (pp. 45–63). National Council of Teachers of Mathematics.

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

Vogel, S., Hoadley, C., Castillo, A. R., & Ascenzi-Moreno, L. (2020). Languages, literacies and literate programming: can we use the latest theories on how bilingual people learn to help us teach computational literacies?. *Computer Science Education, 30*(4), 420-443.

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.

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

Wright, T. S., & Gotwals, A. W. (2017). Supporting kindergartners' science talk in the context of an integrated science and disciplinary literacy curriculum. *The Elementary School Journal, 117*(3), 513-537.

Understanding Barriers to School-Wide Computational Thinking Integration at the Elementary Grades: Lessons From Three Schools

Maya Israel, Ruohan Liu, Wei Yan, University of Florida
Heather Sherwood, Wendy Martin, Education Development Center
Cheri Fancseli, Edgar Rivera-Cash, and *Alexandra Adair,* Research Alliance for New York City Schools

Corresponding Author: Maya Israel, *misrael@coe.ufl.edu*

Abstract

This mixed methods study examined barriers faced by three elementary schools in their approaches to integrating computational thinking (CT) into classroom instruction. Because CT is a relatively new instructional area, limited research guides CT integration, especially in bringing CT to academically, linguistically, and culturally diverse instruction. This study, therefore, examined challenges faced by teachers in bringing CT into their instructional practice. Data included interviews with 11 teachers and surveys of 46 teachers in a large, urban school district. The three schools in the district were part of a CS for All initiative in their school district. Data revealed common challenges: a) Limited CT teaching expertise; b) Limited time for CT integration; c) Lack of CT-specific assessment knowledge and tools; d) Limited pedagogical understanding for meeting students' diverse instructional needs; and e) Low teacher-buy-in for teaching CT. Our data also showed differences across schools in access to classroom implementation infrastructure such as technology and curricular resources, competing administrative priorities, and types of professional development opportunities. This study points to practical implications for supporting integration of CT in elementary contexts. Most notably, it is critical to proactively address these barriers in preservice and in-service teacher preparation as well as in school-wide infrastructure in order to have a sustained CT integration effort.

Introduction

Computational thinking (CT) is increasingly making its way into the elementary grades. It is commonly defined as "an approach to solving problems, designing systems and understanding human behavior that draws on concepts fundamental to computing" (Wing, 2006; 2008, p. 3717). The K-12 Computer Science Framework describes several core practices, with CT comprising four of these practices: Recognizing and defining computational problems; developing and using abstractions; creating computational artifacts; and testing and refining computational artifacts (K-12 CS Framework, 2016). CT can, thus, be considered "both a skill to learn and a way to learn--to create, discover, and make sense of the world, often with computers as extensions and reflections of our minds" (Digital Promise, 2017, p. 21).

In the elementary grades, CT is often integrated into the core content areas (e.g., Fofang et al., 2019; Sherwood et al., 2021). There are three main justifications for this approach to CT integration: (1) practical considerations (e.g., there is not enough time for stand-alone CT instruction), (2) pedagogical rationale (e.g., teaching CT in the context of other disciplines offers a unique way for students to learn, create, and problem solve), and (3) equity (e.g., embedding CT into core academic content provides CT access to all learners (Fofang et al., 2020). Additionally, CT integration is informed by conceptual frameworks of integration used across other content areas. The level of integration can range from disciplinary instruction, where no integration occurs, through trans-disciplinary instruction, where students create new knowledge that transcends the individual disciplines (Vasquez et al., 2013). Despite conceptual and theoretical models offered by Vasquez and others, the term "content integration" is often used as a catch-all for all such instructional experiences (Tytler et al., 2019). Thus, when teachers describe content integration, they may mean a range of approaches. For example,

Sherwood and colleagues (2021) describe CT integration ranging from using academic language that crosses disciplinary areas (e.g., the term decomposition across CS and math instruction) to using complex integrated computer-based activities.

Given this ambiguity, the existing research highlights that teachers feel underprepared to teach CT (Yadav et al., 2018). And, to complicate matters further, literature in science, technology, engineering, and mathematics (STEM) integration suggests instructional concerns such as practical difficulties in creating integrated instructional activities (Hobbs et al., 2018), concerns about inequity between the disciplinary areas (Vasquez et al., 2013), and lack of consensus about what integration looks like during instruction (English, 2016). Given both the ambiguity of what CT integration looks like at the elementary level and the lack of resources to support CT integration, school leaders and teachers must make instructional decisions without guidance from the literature. Additionally, little research exists about the barriers that teachers face when trying to integrate CT into their instructional practice. This literature primarily focuses more on barriers to technology integration (e.g., Ertmer, 1999; Ottenbreit-Leftwich et al., 2018; Pittman & Gaines; 2015). Although this literature provides a framing for considering barriers to CT integration, there is a need for additional research on challenges that teachers face as they attempt CT integration. Therefore, the purpose of this paper is to unpack barriers that teachers face when integrating CT into their elementary instruction and to provide guidelines for addressing those barriers.

METHODS

This mixed methods study employed a concurrent triangulation design (Creswell & Plano-Clark, 2017), with qualitative data serving as the primary source of information which was reinforced and confirmed through quantitative data. Specifically, we engaged in an approach where we triangulated data by source (i.e., teacher interviews and surveys). In such triangulation designs, researchers collect and analyze the qualitative and quantitative data separately and then synthesize that data to interpret the research findings (Creswell & Plano-Clark, 2007). The two research questions that guided this study were: (1) What barriers teachers report related to integrating CT into elementary instruction? (2) Are barriers related to the type of instructional context and professional development provided to the teachers?

Setting and Participants

Three elementary schools in a large urban school district in the Northeast participated in this study. Table 1 provides school demographic information. These schools were recruited because they were involved in a *CSforAll* school-wide initiative in which they integrated CT into instruction.

All three schools received professional development (PD) for integrating CT into their school curriculum. However, each school approached integration in a different manner. School A had a lead teacher who taught all the students in the school for CT integration activities within science instruction. This teacher was the STEM teacher for the school. Although other teachers participated in PD and CT activities, the primary driver of CT integration was the STEM teacher. Their PD consisted of a structured schedule of afterschool workshops with coaching sessions between these workshops. The PD provider initially focused on helping teachers conceptually understand CT but later shifted to more hands-on practical CT integration approaches. School B and C had a more whole-school approach although their PD models differed. In School B, the PD provider went through a scaffolded "I do, we do, you do" approach where they modeled CT instruction, followed by collaboratively teaching CT, and then the teacher took ownership of CT instruction. School B also utilized a structured workshop approach with coaching occurring between these workshops. School C used an embedded coaching model wherein the CT coach spent time with teachers to slowly integrate CT into their instructional practice, beginning with a focus on use of CT academic language and moving towards more authentic unplugged and then plugged CT integration activities. Although School C also had structured workshops, because the coach spent more time in the school, the coach could also collaborate on lesson implementation, provide more frequent feedback, and give more consistent support.

All teachers who participated in the CT integration activities were asked to participate in the research. Of the ones who provided informed consent, all were asked to

Table 1. School Demographic Information

School	Number of Students	% Students with high economic need	% English language learners	% Teachers with 3+ years teaching experience	Teacher participants: Surveys/Interviews
A	500	84.6	33.8	67	8 surveys; 1 interview
B	1600	72.0	29.6	83	12 surveys; 4 interviews
C	1625	86.5	31.4	71	26 surveys; 6 interviews

complete the surveys. Given the varying approaches to integration, a different number of teachers in each school were selected to participate in interviews. The STEM teacher was interviewed in School A as she was the primary driver of CT integration at her school. In school B, there were only two teachers per grade level who integrated CT into instruction so these two teachers per grade level were interviewed (4 teachers). In School 3, which was the largest, two teachers per grade-level were randomly chosen to be interviewed (6 teachers).

CT Integration Approaches. Across the schools, the teachers had different approaches to integration based on the approaches provided by the PD providers. In School A, the STEM teacher integrated CT into science instruction across all the grades in the school. In Schools B and C, the teachers integrated CT primarily in literacy and math instruction as these were the areas of priority for the schools. The PD providers encouraged the teachers to use academic language associated with computational thinking (e.g., sequencing, decomposition, debugging) as well as a combination of unplugged and plugged activities.

Data Collection. Data included teacher surveys and semi-structured interviews across all three schools. Interviews and surveys took place after the initial year of CT integration. This way, the teachers were able to describe the barriers they faced during the process. The interview questions were designed to be open ended and allow teachers to describe their experiences with teaching CT and the barriers that they encountered. Initial questions focused on asking the teachers to share their experiences with teaching CT in general rather than on barriers. In this way, we avoided leading the teachers to discuss barriers. If they described barrires, the semi-structured interview protocol allowed for follow-up questions specific to barriers. After this set of questions, the teachers were asked one question specific to any barriers they encountered when implementing lessons or activities to teach CT.

Surveys included 12 five-point Likert-scale items related to anticipated barriers.

This item was worded as, "To what extent do you anticipate the following challenges will interfere with the implementation of CS and CT in your classroom? Example items included lack of curricular materials, lack of administrative support, lack of student interest in CS and CT. This scale ranged from *not at all to a great extent*. Cronbach's Alpha value for the survey items is 93%, which indicates high internal consistency of the survey items.

Data Analysis. Qualitative data analysis involved an inductive coding using a constant comparative approach (Glaser, 1965) within the Dedoose qualitative coding software. The research team developed a structured coding scheme that involved the code name, operational definition, and code example (see Table 2).

All the interview data was coded by two researchers in three phases. Phase 1 involved collaboratively coding teacher interviews and developing the code book. In phase 2, once the code book was established, data was divided among the coders for individual first-pass analysis. Data that was analyzed individually was entered into a table and checked for agreement with the other coders in the second-pass analysis. When differences emerged, codes were compared and discussed for agreement. When necessary, the code scheme was revised for clarification and data was recoded based on the new coding criteria. All data was checked again for accuracy to ensure that analysis reflected any changes that resulted from the constant-comparative process. Analysis of the surveys involved frequency counts of the Likert scale items to ascertain the barriers that were identified as most prevalent among the survey items. Finally, interview and survey data were compared as a means of triangulating data sources.

RESULTS

Interview Results

Teachers across all three schools described five primary barriers to CT integration (See Table 3):

Table 2. Example Codes, Operational Definitions, and Example Interview Quotes

Example Code	Definition	Example quotes from interviews
CS/CT is a new instructional area	Teacher describes the process of learning how to teach CS/CT	"I'm kind of like learning and I'm still learning the Scratch skills myself and there's a lot of times that I end up saying to the kids, like, "You probably know how to do it better than I do, so that's okay," like we're learning together".
Integration of CT into content areas	Teacher describes bringing CT concepts and practices into the content areas	"What the best way is to put these into different subject areas and to integrate it….I think it's something that you teach with the concept that you're teaching…How do we mix it into the curriculum so that it doesn't become an extra thing".

Table 3. Barrier Categories from the Teacher Interviews

Barrier	# Teachers (n=11)	Illustrative Quotes
Limited knowledge related to how to teach CT in their classrooms.	8	"I think the biggest challenge was it was a very steep learning curve... So as I was teaching the students and I'm still learning the Scratch skills myself."–School C
Finding time for CT in the school day	6	"In the beginning, you're thinking about time. I don't have time...Even in these 10-minute engaging activities, where does that come from the day?–School B
Lack of CT assessment knowledge and tools	5	"I think the biggest challenge if I had to root it in CT is that there's still not concrete performance indicators."–School A
Challenges with meeting students' diverse needs in computing	5	"I think just with the different levels of students, I think it's hard for some classes. Just that they don't have the foundations from the previous grade and they come already into second grade lacking that."–School C
Limited teacher buy-in about teaching CS/CT	5	"I think the biggest challenge would be having teachers that are willing to implement something new. I think that's the hardest part about, you know, introducing anything."–School B

Limited knowledge related to how to teach CT in their classrooms. When the teachers described lack of knowledge of CT, they did so both from the perspective of having a limited understanding of CT concepts (e.g., decomposition, abstraction) and how to teach these concepts in the context of integrating CT into academic content areas such as literacy, math, or science. Teachers' lack of expertise encompassed three interrelated areas: CT concepts, pedagogical approaches for introducing CT to their learners, and integration of CT into the core academic content areas. The teachers, thus, indicated that they taught CT integrated lessons in an instructionally ambiguous situation wherein they were unsure what and how to teach CT in an integrated manner.

Finding time to teach CT. Teachers indicated that a major barrier to CT integration was finding time to teach CT during the school day when there was already limited time for the core content areas, primarily reading and mathematics. Teachers also described difficulty finding time learn the content well enough to teach it to their learners.

Lack of knowledge of and access to CT assessment. This barrier specifically addressed how teachers would know whether the students learned the CT concepts that were being introduced in class. They indicated that they had no performance indicators, no ways of understanding how students can apply skills learned to new problems, or no understanding of what assessments might look like in the context of CT integration.

Challenges with meeting the needs of students with a wide range of learning needs. Teachers indicated that their student population is diverse and includes students who are emergent bilingual learners, students with disabilities, and students who were struggling for other reasons. They expressed a tension between providing basic core academic instruction alongside CT as well as in how to provide enriching CT instruction to all their learners.

Limited teacher buy-in. Although buy-in emerged as a barrier to implementation, there were a range of reasons for this limited buy-in. The most common for this was unease with the amount of work it would require to introduce CT into the content areas. The teachers described limited energy, resources, and will for implementing something new, given all of their instructional demands. Another reason expressed by the teachers was skepticism about whether CT integration would result in academic benefits for their learners. Simply put, there was not enough research about the efficacy of CT integration to make it worthwhile for some teachers to invest time and energy to introduce it into their teaching.

Survey Results

Survey results showcased similar findings to those in the interviews. Barriers reported in the surveys included lack of time to implement CT (n=26) and lack of available curricular materials (n=17). In addition to these barriers that appeared in both the interviews and surveys, the survey data also revealed barriers not described in the interviews: Lack of alignment between CT priorities and teacher evaluation (n=17); competing priorities between CT and other school priorities (n=33). Figure 1 provides the list of areas the teachers indicated on the survey were barriers to a moderate or great extent. As Figure 1 also shows, the one area where teachers across all schools indicated that student interest in CS and CT was not a barrier.

Differences among the schools. Although teachers across schools indicated similar barriers, because of different PD and integration approaches, some differences emerged.

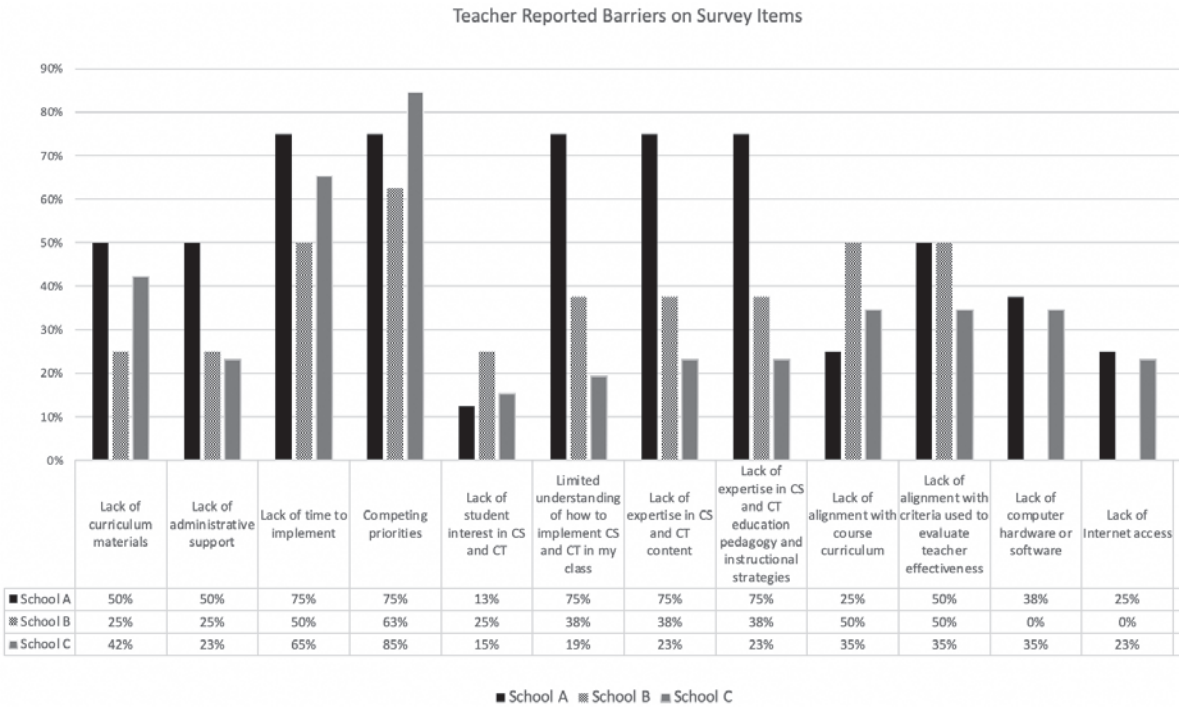


Figure 1. Percentage of Teachers Reporting Factors that were Moderate or Great Barriers

School A. The primary CT integration driver in School A was the STEM teacher. When interviewed, she indicated that although other teachers in the school attended PD, the other teachers did not develop or implement the CT lessons without her involvement. Thus, the STEM teacher had CT integration responsibilities beyond her own classroom CT activities. When the STEM teacher described time, it was time to work with the PD provider to design instruction for both her own class as well as time for implementing CT instruction in the other classrooms. She stated, "I think that time is the biggest factor in developing a new curriculum [at all grade levels]." School A also had a higher proportion of teachers who indicated limited understanding of CS and CT content, as well as how to implement CS and CT in their classrooms, including which pedagogical approaches are effective in teaching integrated CS and CT. Given that these teachers were not primarily responsible for CT instruction, this finding was unsurprising.

School B. The teachers in School B primarily described lack of time as finding time in the school day to teach CT. The PD providers helped to co-construct lessons with the teachers, so the teachers had to find time to both work with the PD providers and teach the lessons. The school administrators allowed teachers and PD providers to use instructional planning time for co-planning. One teacher in School B explained, "Finding that balance of how you can integrate the CT and still do everything that you need to do throughout the day." Teachers in School B had the lowest proportion of teachers who stated that they lacked curricular materials and none of the teachers in this school indicated a lack of computer hardware, software, or internet access. They had the most access to tools and technologies, compared to the other schools. In interviews, they stated that they had access to all necessary technologies to implement CT integration.

School C. School C teachers described lack of time as a barrier related to finding time during the school day, but they did not indicate planning time as a major barrier as their PD provider provided them with lesson plans that could be implemented without a great deal of development work. The teachers explained that the administrators allowed them to use existing planning time for CT integration planning and working with their PD providers. Compared to Schools A and B, teachers in School C had the lowest proportion of teachers who stated that they had a limited understanding CS and CT as well as pedagogical approaches and ways of integrating CS and CT into their own instructional settings. This finding was unsurprising as they had the most access to PD as part of their PD implementation initiative.

Discussion

This study highlights the common barriers faced by elementary teachers who were integrating CT into their instruction. Although all three schools offered PD and had resources for implementing CT, all the teachers indicated that this work was difficult and time consuming, similar to findings by Yadav and colleagues (2016). Although we

expected a great deal of variation among the barriers reported by teachers at the different schools, there were more commonalities than differences. The differences that emerged could be aligned with the different PD approaches. For example, School C had the most intensive PD and coaching approach, so teachers indicated greater understanding of CT integration.

Previous research on content integration (e.g., Tytler et al., 2019) and specifically integration of CT into instruction (e.g., Yadav, 2018) highlight both the ambiguity of what is meant by content integration and the difficulty that teachers face in integrating CT into their classroom instruction. This challenge was evident in teachers' discussion of their lack of expertise related to CT integration in three ways: (1) CT understanding, (2) understanding of effective pedagogies for teaching CT, and (3) ways to bring CT into their core academic areas. Even with robust professional development, teachers struggled with identifying the connections between their core content instructional goals and CT concepts and practices. These CT integration challenges must also be viewed in light of other challenges faced by elementary teachers including teaching multiple subjects, classroom management, and insufficient planning time (e.g., Conway, 2001; Feiman-Nemser et al., 1999; Lortie, 1975; Veenman, 1984).

Barriers reported by the teachers were consistent with first and second order barriers to technology implementation reported in the literature (Ertmer, 1999). First order barriers (i.e., institutional barriers) are extrinsic to the teachers and include both lack of time to teach and plan integrated CT instruction as well as the competing priorities that the teachers indicated on the surveys. These barriers can result in significant challenges for teachers because of lack of resources, support, and other aspects outside of the teachers' control. Second order barriers (i.e., personal barriers) relate more to teachers' underlying beliefs about their own capacity to teach CT instruction as well as the importance of such instruction. In this study, the teachers described a lack of content, pedagogy, and assessment knowledge as well as lack of buy-in for teaching CS and CT. Although on the surface, the first order barriers may seem like the more challenging ones to address, it may actually be these second order barriers that are the more difficult to address given that they require teacher learning and change (Ertmer, 2005). Thus, it is important to consider teacher learning and beliefs along a continuum so that what is defined as successful CT integration for a teacher new to CT will look different from CT integration for a teacher who has prior CT teaching experience.

Although the teachers did not rank lack of administrative support as a major barrier, the types of barriers they did report were those that administrators can influence. For example, survey results indicated that competing priorities and time were both major barriers. A team-based approach where district and school-based leadership engage with teachers can help address some of these barriers (DeLyser et al., 2018). For example, the SCRIPTS process that the CS for All Consortium facilitates may assist district and school leadership towards addressing both the first and second order barriers found in this study.

It is interesting that some barriers reported in the survey were not reported in interviews. For example, 17 teachers responded on the survey that there was a lack of alignment between CT priorities and criteria used by administrators to evaluate teacher effectiveness. Similarly, 12 teachers reported a lack of administrative support as a barrier. However, this was not a major theme in the interviews. It could be that the teachers did not feel comfortable sharing concerns related to their administration with the researchers in the interviews, but they were comfortable reporting these results in an anonymous survey. Future research should, therefore, address barriers in a manner that maintains teacher anonymity during the research process. Lastly, all the teachers in this study had access to ongoing support and PD to develop their CT integration knowledge and skills. Despite this support, they still reported multiple barriers. However, in many cases, teachers are faced with integrating CT into their instruction on their own and taking on the leadership for both increasing their own capacity and providing professional support to other teachers in their buildings. Given the findings of this study, especially as related to lack of time and knowledge, this CT integration work should not be placed on one or two teachers, but should take on a systems approach (DeLyser et al., 2020).

Limitations. Several limitations should be acknowledged within this study. First, this study took place within one district that had a range of CS and CT instructional resources available to the teachers. Although PD differed among the schools, all teachers had access to PD and the necessary tools for implementing CT. These conditions are not likely generalizable to smaller school districts or those without a K-12 CS initiative. Second, this study did not triangulate findings about barriers with the data from administrator interviews. Although administrators were involved in the CT integration within their schools by helping to develop the vision and participating in leadership decision-making, they were not used as a specific data source related to the barriers reported by the teachers in order to honor the confidentiality of the participants. Future research should explore how administrators view CT initiatives in light of the competing priorities described by the teachers. Lastly, this study did not investigate the extent to which previous experience with teaching CT influenced the teachers' perceived barriers to implementing CT integration. Given that disciplinary experience with CT likely shapes level-two barriers, this connection should be explored further.

Implications for Practice: Suggestions for Teachers New to CT Integration

We propose several strategies for addressing barriers to CT integration based on the findings of this study. First, it is essential to build a school culture of CT integration that includes a plan for CT integration, building time into the work day for planning, and promoting ongoing professional learning (Israel et al., 2015). If the school provides common planning time, teachers can use that time to develop lesson plans that infuse CT into curricula, share instructional strategies, and reflect on teaching experience. Next, joining and participating in face-to-face and online professional learning communities (Han & Liu, 2021) can enhance teachers' engagement in CT learning and provide both local and remote colleagues to support learning, provide resources, and help address implementation barriers. There are a number of communities that can help CS teachers feel less isolated, provide resources and strategies, and address common barriers. These communities include: (1) the Computer Science Teachers Association (CSTA) national and state divisions (https://www.csteachers.org/), (2) the CSTA K-8 Facebook page, and (3) Twitter chats such as #CSK8 and #CSforAll. Next, as it is important to consider a systems-approach to proactively plan for introducing CT into classroom contexts, supporting teachers in gaining the skills necessary to do so, and then sustaining CT instructional initatives. Tools such as the CS for All SCRIPT (Strategic CSforALL Planning Tool for School Districts) program, which is a framework that guides teams through visioning, goal setting, and planning for CS education implementation, can provide the necessary guidance. Lastly, many elementary teachers are novice to CT, so it is important to find resources online. There is a growing list of online resources for CT integration. Below are a couple integration-specific resources:

- Project GUTS elementary modules (https://teacherswithguts.org/welcome) provide integrated CT and science instruction.
- EverydayCS Action Fractions (http://everydaycomputing.org/) is a Scratch-based 3rd and 4th grade integrated fractions and CT curriculum.

References

Angevine, C., Cator, K., Roschelle, J., Thomas, S. A., Waite, C., & Weisgrau, J. (2017). *Computational Thinking for a Computational World*. Digital Promise. Retrieved from https://digitalpromise.org/wp-content/uploads/2017/12/dp-comp-thinking-v1r5.pdf

Conway, M. A. (2001). Sensory–perceptual episodic memory and its context: Autobiographical memory. *Philosophical Transactions of the Royal Society of London*. Series B: Biological Sciences, 356(1413), 1375-1384.

Creswell, J. W., & Clark, V. L. P. (2017). *Designing and conducting mixed methods research*. Sage publications.

DeLyser, L. A., & Wright, L. (2018, February). Creating a Landscape of K-12 CS Curriculum. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education* (pp. 1098-1098).

DeLyser, L. A., Wright, L., Wortel-London, S., & Bora, A. (2020, February). Evaluating A Systems Approach to District CS Education Implementation. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education* (pp. 1120-1126).

English, L. D. (2016). STEM education K-12: Perspectives on integration. *International Journal of STEM education, 3*(1), 1-8.

Ertmer, P. A. (1999). Addressing first-and second-order barriers to change: Strategies for technology integration. *Educational technology research and development, 47*(4), 47-61.

Feiman-Nemser, S., Schwille, S., Carver, C., & Yusko, B. (1999). A Conceptual Review of Literature on New Teacher Induction. Retrieved from https://files.eric.ed.gov/fulltext/ED449147.pdf .

Fofang, J. S., Weintrop, D., Walton, M., Elby, A., & Walkoe, J. (2020). Mutually Supportive Mathematics and Computational Thinking in a Fourth-Grade Classroom. In Gresalfi, M. and Horn, I. S. (Eds.), The Interdisciplinarity of the Learning Sciences, *14th International Conference of the Learning Sciences (ICLS) 2020*, Volume 3 (pp. 1389-1396). Nashville, Tennessee: International Society of the Learning Sciences.

Glaser, B. G. (1965). The constant comparative method of qualitative analysis. *Social problems, 12*(4), 436-445.

Han, S., & Liu, M. (2021, March). Empowering Teachers through Professional Development Program and Online Learning Community. In Society for Information Technology & Teacher Education International Conference (pp. 1236-1241). Association for the Advancement of Computing in Education (AACE).

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.

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

Lortie, D. (1975). Schoolteacher: A sociological study. Chicago, IL: Univ of Chicago Press.

Ottenbreit-Leftwich, A., Liao, J. Y. C., Sadik, O., & Ertmer, P. (2018). Evolution of teachers' technology integration knowledge, beliefs, and practices: How can we support beginning teachers use of technology? *Journal of Research on Technology in Education, 50*(4), 282-304.

Pittman, T., & Gaines, T. (2015). Technology integration in third, fourth and fifth grade classrooms in a Florida school district. *Educational Technology Research and Development, 63*(4), 539-554.

Sherwood, H., Yan, W., Liu, R., Martin, W., Adair, A., Fancsali, C., ... & Israel, M. (2021, March). Diverse Approaches to School-wide Computational Thinking Integration at the Elementary Grades: A Cross-case Analysis. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education* (pp. 253-259).

Tytler, R., Prain, V., & Hobbs, L. (2019). Rethinking disciplinary links in interdisciplinary STEM learning: A temporal model. *Research in Science Education*, 1-19.

Vasquez, J., Sneider, C., & Comer, M. (2013). STEM lesson essentials, grades 3–8: integrating science, technology, engineering, and mathematics. Portsmouth, NH: Heinemann.

Veenman, S. (1984). Perceived problems of beginning teachers. *Review of Educational Research, 54*(2), 143-178.

Yadav, A., Gretter, S., Hambrusch, S., & Sands, P. (2016). Expanding computer science education in schools: understanding teacher experiences and challenges. *Computer Science Education, 26*(4), 235-254.

Yadav, A., Krist, C., Good, J., & Caeli, E. N. (2018). Computational thinking in elementary classrooms: measuring teacher understanding of computational ideas for teaching science. *Computer Science Education, 28*(4), 371-400.

Strengthening Early STEM Learning by Integrating CT into Science and Math Activities at Home

Shuchi Grover, Looking Glass Ventures
Ximena Dominguez, Tiffany Leones, Danae Kamdar, Digital Promise
Phil Vahey, Houghton Mifflin Harcourt, ***and Sara Gracely,*** SRI International

Corresponding Author: Shuchi Grover, *shuchig@cs.stanford.edu*

Abstract

While understanding in the field of how CT can be used in early childhood is limited, current CT definitions include skills and practices that align with early mathematics and science learning goals outlined in national frameworks (e.g., Head Start Early Learning Outcomes Framework) and state standards. In order to understand which elements of CT align with the abilities and interests of young children and how they can be integrated with early science and math experiences in a mutually supportive manner, we partnered with preschool teachers and families to co-design and pilot test hands-on (unplugged) and digital activities for classrooms and homes.

Our collaborative research identified the following CT skills as productive starting points for our co-design work: (1) problem decomposition; (2) algorithmic thinking; (3) abstraction; and (4) testing and debugging. This paper describes our approach to operationalizing CT for early learning and our empirical research around activities designed to understand how CT can be linked with math and science to create powerful learning experiences for preschool learners. Our work involves actively fostering a home-school connection for promoting CT and prioritized designing for activities that fit the ecology of preschool classrooms and homes (with special attention to family activities). With a view to designing equity-oriented experiences, we partnered with preschools serving historically underserved communities, and centered families' funds of knowledge. This paper focuses specifically on the home component of the program and shares data and analyses about children's and parent's experiences at home—which activities were more successful and resonated with children and families, and which specific synergies emerged between CT skills, math and science concepts and practices.

Our findings highlight the promise of introducing early CT to support early learning, and especially involving families in the process. Results from our research also identified challenges that should be addressed in future iterations of this design research We believe our family connection activities are not only a unique part of the research but also an exemplar of what should be an essential piece of STEM education for young learners.

Introduction

Five years ago, the 'Computer Science for All' initiative was launched to "empower all American students to be equipped with the computational thinking skills they need [and] to be active citizens in our technology-driven world" (White House, 2016). Although much of the early attention on computational thinking (CT) in K-12 has focused on resources for secondary school students, new tools and activities directed to elementary children have also proliferated in recent years with the creation of environments focusing on programming (e.g., Scratch Jr., Kodable, HopScotch); activity sequences (e.g., 'Hour of Code' by Code.org); apps (e.g., Daisy the Dinosaur); and tangible programmable robots (e.g., Bee-Bot, Dash & Dot). Most of these approaches seek to introduce children to fundamental ideas of algorithmic thinking and its elements—sequence, conditional thinking and repetition. Few, however, have examined CT at the preschool level, and especially the synergistic—or mutually supportive—integration of CT learning with other early childhood learning domains. Studies have shown that children's experiences in preschool math, English, and science can strongly predict later academic success, and not just in these subjects (Duncan et al., 2007). As

educators explore how CT can be integrated meaningfully to support learning, it is important to investigate how elements of CT best align with the abilities and interests of preschool children. In what contexts can CT be promoted in a meaningful and consequential way? How can CT activities be integrated into common preschool classroom activities and home learning experiences to strengthen young children's STEM learning, especially for children in historically underserved communities? These questions provided the motivation for this research.

This paper describes our research activities that aimed to understand how CT can be linked with math and science to create powerful learning experiences for preschool learners. Our research centers equity; it involves (a) purposefully partnering with preschools serving culturally and linguistically diverse families in low-income communities (Nasir et al., 2020) and (b) co-designing with and drawing from families' funds of knowledge (Moll, 2015; Moll et al., 1992). We start by framing our work in the dual contexts of early STEM learning research, including evidence highlighting the importance of home-school connections, as well current CT frameworks K-12 learning. We, then, briefly discuss how we operationalize CT for early learning and describe our approach to iteratively co-design activities (Penuel, Roschelle, Shechtman, 2007), which involved bringing together public preschool teachers, families, curriculum and media designers, and CS education and early learning researchers. A series of design-based research activities (DBR; Sandoval & Bell, 2004), including observations, surveys, interviews, and assessments, aimed to examine how CT activities at home support young children's learning. In this paper, we discuss findings from a culminating field study involving public preschool classrooms and homes and discuss implications and learnings for the field as well as our ongoing and future efforts.

The Early Learning Context

The early childhood education field is increasingly aware of the need for and value of promoting early learning in STEM (Schweingruber, Duschl, & Shouse, 2007; U.S. DHHS, 2015). This recognition emerged from research determining that (1) young children are interested in and have a right to engage in mathematics and science as a way of exploring and understanding the world around them, (2) learning mathematics and science often includes exploratory and play-based activities that resonate with developmentally appropriate preschool practices, (3) scaffolding young children's mathematics and science learning can promote the reasoning skills and learning dispositions crucial for later school success (Clements & Sarama, 2014; Gelman, & Brenneman, 2004), and (4) promoting mathematics and science can result in increased interest in and better preparation for later STEM learning (Leibham, Alexander, Johnson, 2013). Given the persistent problems of inequity in STEM, efforts to promote integrated STEM learning are especially needed in public preschool programs serving children from historically underserved communities. Recent advances in our understanding of the building blocks of CT and of the capabilities of young learners make the integration of CT with early STEM activities ripe for exploration.

Some researchers have explored non-programming activities for building CT skills and practices in preschool, but evidence evaluating children's learning from the activities designed is limited. Mittermeir (2013) found that a small group of preschoolers could articulate algorithms for sorting tasks. Calderon, et al. (2015) shared a prototype idea for pattern recognition among children ages 3 to 5. Horn, AlSulaiman, and Koh (2013) developed an interactive storybook that leveraged family language and literacy practices to help preschool and elementary students apply ideas about sequences and loops. Emerging research indicates that preschoolers are able to learn CT skills such as sequence, modularity, and debugging in the context of simple, engaging hands-on classroom math activities (Lavigne et al., 2020).

Our work involves actively fostering a home-school connection for promoting CT and prioritized designing for activities that fit the ecology of preschool classrooms (e.g., circle time, small group activities, learning centers) and homes (e.g., board games, paired/family activities). This draws on the strong evidence in early learning literature about the positive impact of family involvement in early STEM learning. Based on a meta-analysis of 46 studies, Ma et al. (2016) found a strong and positive correlation between learning outcomes and parental involvement, and that the role of parents (family involvement) was more important for STEM learning than the role of schools and communities (partnership development). The authors argued that to develop a strong relationship between learning outcomes and parental involvement, home supervision, behavioral involvement, and home-school connection were the keys from family involvement. Additionally, scholars studying equity in computing education more broadly suggest that connecting computational experiences to family and community is part of culturally responsive computing pedagogy (Scott, Sheridan, & Clark, 2015) and is an equitable approach to teaching computing (Pinkard et al., 2020). Roshan, Jacobs, Dye, and DiSalvo (2014) argued that parents in economically depressed communities struggle to negotiate what roles they can play in their children's computational learning experiences, and how they can help their children access computing-related learning resources. We believe our family connection activities are not only a unique part of the research but also an exemplar of what should be an essential piece of STEM education for young learners.

Framing Computational Thinking

CT has been described as a composite set of problem-solving skills associated with computer science but with overlapping roots in mathematics and engineering. Wing (2006) described CT as a universal skill for all in today's world, as computing has become a pervasive part of life. Researchers focusing on primary and secondary computer science education have further articulated the problem-solving strategies that comprise CT as including algorithmic thinking, decomposition, abstraction, pattern recognition, generalization, systematic error detection and debugging, and evaluation of solutions (Grover & Pea, 2013, 2018). More recently, Dong et al. (2019) synthesized various articulations of CT, including those catering to STEM integration contexts (e.g., Weintrop et al., 2016) to propose PRADA—Pattern Recognition (observing and identifying patterns, trends, and regularities in data, processes, or problems), Abstraction (identifying the general principles and properties that are important and relevant to the problem), Decomposition (breaking down data, processes, or problems into meaningful smaller, manageable parts), and Algorithms (developing step by step instructions for solving [a problem] and similar problems).

Although research on CT in early childhood is sparse, current CT definitions resonate with school readiness goals delineated by preschool programs and policies. For example, the Head Start Outcome Framework's Reasoning and Problem-Solving subdomain indicates that young children need to learn to use a variety of problem-solving strategies, and reason and plan ahead to solve problems (DHHS, 2015). Research that examines how these CT skills align with the abilities and interests of young learners and uniquely support their school readiness is thus timely and needed. Our work on integrating CT into preschool learners' STEM activities drew on these frameworks, but also required that we find age-appropriate ways of accomplishing our goals and identifying productive points of synergy among elements of CT and early STEM learning goals. Through this project we are developing a deeper understanding of what CT looks like in its simplest form, and how it can enrich math and science for preschoolers.

METHODS

The research presented in this paper is part of a broader effort that aims to examine how we can bring CT into early learners' lives through curricular activities that integrate CT with familiar preschool science and math activities and play in ways that fit seamlessly into classroom as well as home routines; and familiarizing teachers and parents/caregivers on what CT is and preparing them to implement the activities at school and home, respectively. ***This paper focuses on the home component of the program and shares findings about children's and parent's experiences at home.***

Operationalizing CT, Connecting to Early Math and Science, and Co-Designing Activities

Early co-design work helped identify target content and initiate the design of the learning blueprints that in turn helped curricular design of activities. As part of the initial stages of co-design (Figure 1), our team identified the following CT skills as productive starting points for our work: (1) problem decomposition; (2) algorithmic thinking; (3) abstraction; and (4) testing and debugging. These were identified as promising entry points for exploring integration with other STEM areas by the co-design team and project advisors based on their own prior research and design experiences in early math and science (Presser et al., 2019; Vahey et al., 2018), existing research evidence (described above), and alignment with current early childhood standards (e.g., Head Start Early Learning Outcomes Framework and state standards such as California's Preschool Learning Foundations).

A 'learning blueprint' (Vahey et al., 2018) was generated to list developmentally appropriate learning goals for each of the identified CT skills and delineate possible connections to early mathematics and science. For instance, *algorithmic thinking* was tagged in relation to counting and visual spatial skills common in navigational activities, *abstraction* activities were tagged as possibly related to science practices such as comparing, and sorting based on key characteristics and *debugging* seen as tying to scientific experimentation. Additionally, we aimed to give familiar children's activities a goal- or problem solving- orientation. Children would not merely sort objects, but sort and label in ways that help them build something *more efficiently*. For example, an object with red blocks would be more efficiently built if blocks were sorted by color and building an object with square blocks would benefit from sorting by shape.

Prototype hands-on activities were pilot tested in two classrooms and three homes to inform next DBR iterations. 22 classroom and 6 home hands-on activities emerged from revisions. Table 1 shows a list of sample activities, target CT skills, and mappings to math and science[1]. Two digital games were designed and iterated on (and are now freely available for iOS and Google mobile devices) for use in home and school. In *City Walk* (Figure 2), children create a sequence of instructions to help their robot navigate a route to deliver gifts to friends around town. Through appropriate scaffolding, the app progressively introduces more complex tasks and provides visual and audio feedback to help children learn how to debug errors. In *Better Building* (Figure 2), children sort and label groups of blocks based on characteristics such as shape, color, and size, to help the robot build structures more efficiently.

[1] For more details on these activities, including lesson plans, please visit https://digitalpromise.org/initiative/learning-sciences/preschool-computational-thinking/activities/

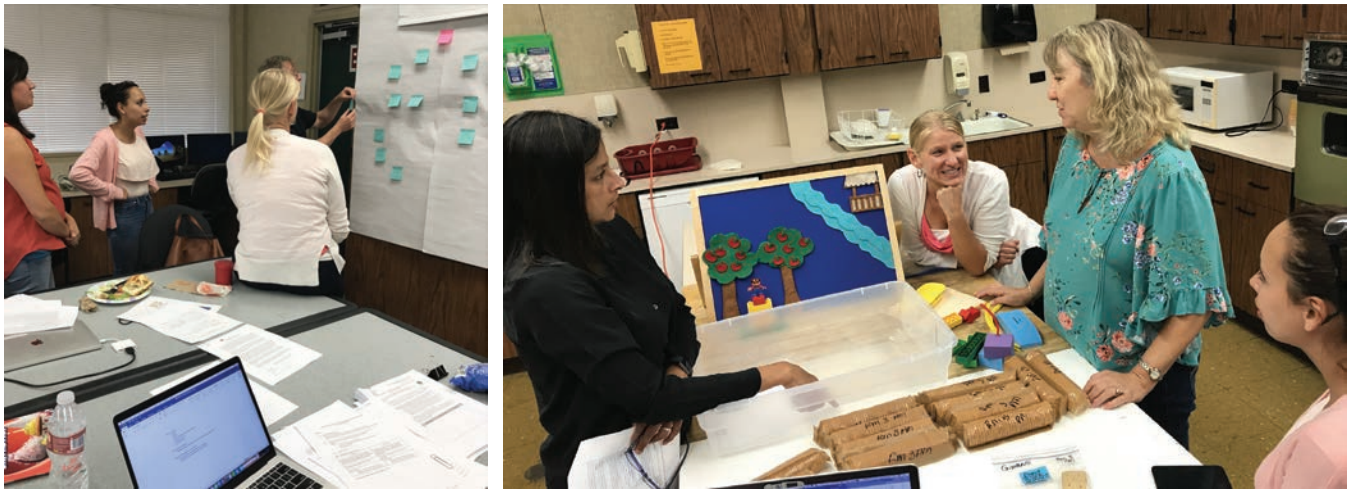


Figure 1. Activity co-design with parents/teachers and caregivers

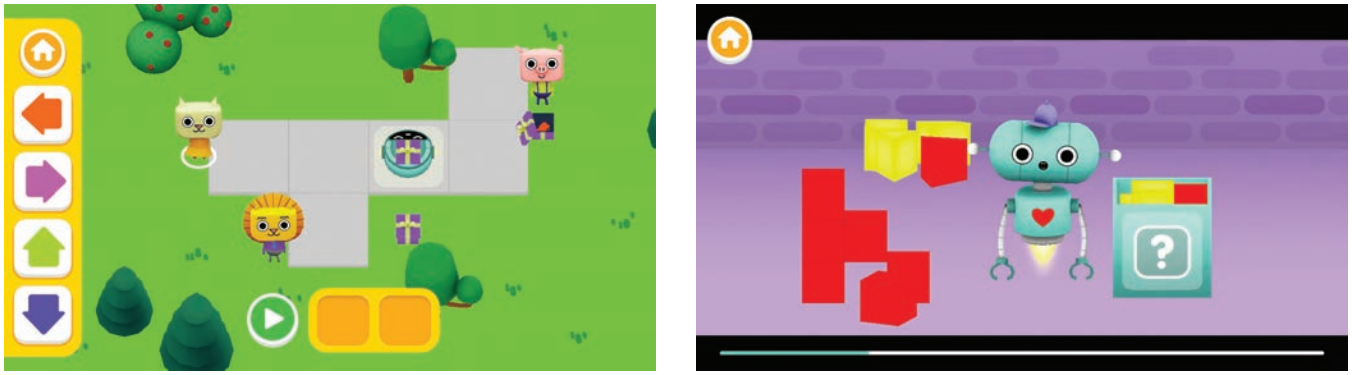


Figure 2. Digital apps/games for : (Left) City Walk (for algorithmic thinking) and (Right) Better Building (for abstraction)

The 6 hands-on home activities (listed in the shaded rows in Table 1) are briefly described below:

A. **Playdough Workshop:** In this activity, children make a playdough creation, such as a rainbow or pizza (see Figure 3) and engage with problem decomposition by identifying the smaller parts they need to make in order to tackle their big creation.

B. **Cereal Necklace:** This activity centered on children learning to identify and follow a sequence of steps (algorithms) to make a necklace out of cereal (or pasta).

C. **Grocery Store Trip:** The goal of this activity was to help families plan what they need at the grocery store and sort items on their grocery list by category (such as fruits, dairy, vegetables, cereal) to make the search at the store easier.

D. **Getting Ready for Bed:** As children create their own book to tell the story of their bedtime routine, they learn to identify the big task and the smaller, sub-tasks.

E. **Playing with Dice:** In this activity, children learn to create instruction codes with loops to play a fun game with numbers and activity dice. Children paired the number on the dice to repetition of actions such as clapping, jumping, etc.

F. **Robot in the City:** As an unplugged complement to *City Walk*, this activity involves children practicing with directional arrows in order to follow a short sequence of directions on a small city map printed as a 3x3 grid.

Figure 3. Problem decomposition through creations in Playdough Workshop

Additionally, we developed a playful task-based assessment to measure CT learning. However, its use in this research was mainly to pilot the assessment while also gathering data for iterative refinements. Sample assessment items are described in Table 2.

Table 1. Sample Activities, formats, and locus along with CT/Science/Math alignments (Blue rows indicate activities designed for families and homes)

Sample Activities *Format/Locus*	CT Skills	Mathematics Concepts/Skills	Science Content/Skills/Practices
City Walk *Digital/Home & School*	Algorithms Debugging	Spatial reasoning/ visual spatial	
Better Building *Digital/Home & School*	Abstraction	Recognizing shapes	Practices: Classifying & sorting, Comparing & contrasting
Carmella's Apple Store *Unplugged/School*	Problem Decomposition, Testing/Debugging	Measurement Counting, Cardinality	Sink and float, ramps & pathways. Practices: Observation, Developing & planning investigations; Cause & effect
Robot in the City *Unplugged/Home & School*	Algorithms (Sequences); Debugging	Spatial reasoning/ visual spatial	
Playing with Dice *Unplugged/Home & School*	Algorithms (Loops)	Number sense (quantity)	
Cereal Necklace *Unplugged/Home*	Algorithms	number, first, next, Patterns	
Playdough Workshop *Unplugged/Home & School*	Problem Decomposition	Shapes, Counting, Cardinality	Practices: Observing & describing
Grocery Store Trip *Unplugged/Home*	Abstraction, Pattern Recognition	Counting	Practices: Observing & describing, Classifying & sorting, Comparing & contrasting, Content: Food/Nutrition; Animals classification
Our Very Own Zoo *Unplugged/School*			
Getting Ready for Bed *Unplugged/Home*	Problem Decomposition	Counting, Cardinality	Content: Hygiene, nutrition
Getting Ready for School *Unplugged/School*			

Table 2. Sample Assessment Items

Item	CT Skill/Practice	Description	Response Format
3	Algorithms (sequence)	Interpret (or follow) code to navigate a small map. Code involves pictures.	Verbal response, point or indicate
5	Algorithms (repetition/looping)	Generate code that includes a loop. Jump three (3) times.	Tangible Manipulative placement
7	Abstraction (sorting)	Sort toy vehicles for a given purpose. (two variables - color and vehicle type)	Tangible Manipulative placement
12	Testing and Debugging	Debug a mistake (made by assessor) in navigation of a small map.	Verbal response, point or indicate
13	Abstraction (labeling)	Label sorted groups of blocks for a given purpose. (one variable - color)	Tangible Manipulative placement
15	Problem Decomposition	Decompose steps to build a simple structure with blocks.	Verbal response, point or indicate

Field Study

The activities and resources developed throughout the project were examined in a small quasi-experimental field study over a six-week period conducted in both classrooms and homes. In this paper, our data and findings focus on examining implementation in homes, with a view to understanding (a) how we can engage families in such activities, and (b) parent/caregiver experiences with CT and its integration with math and science. *This paper shares the research guided by the specific research question:* ***How can we integrate CT for preschoolers into activities at home and what are parent/caregivers' experiences with such integration?***

Sample. A total of 7 public preschool classrooms consented to be part of the field study; 5 classrooms were assigned to the intervention condition and 2 classrooms were assigned to a comparison condition. A total of 2 families (consisting of at least one parent/caregiver and preschool aged child) in each of the intervention classrooms were randomly selected to be part of the home intervention (N=10 families; 4 based in California and 6 in Virginia). All families belonged to minority ethnic groups in the US– 6 were Hispanic/Latino, 3 were Black and 1 was Asian. Parents'/caregivers' highest level of education was either high school or college (Table 3).

Resources and Supports for Families. In preparation for the field study, the team also prepared professional development materials for two meetings which included documents, an infographic video describing CT skills (https://vimeo.com/561877371/9959d88a08) with relatable examples (e.g. we engage in algorithmic thinking when following a recipe with clear, sequential instructions), and a walkthrough to introduce hands-on and digital activities (see examples in Table 1) that families were asked to try out at home.

Data Measures. The following data were collected:

- *Home observations.* We conducted home observations with detailed notetaking (Emerson, Fretz, & Shaw, 2011). Researchers met with each family three times over the course of the field study (once for each CT unit: abstraction, problem decomposition, algorithmic thinking). During the family meetings, researchers took notes on families' experiences with the hands-on activities and digital apps, and documented their feedback. Family meetings in VA took place at the preschool. In CA, family meetings took place in a variety of settings: outdoor space on the school premise (one family); home (one family); and local library (two families).
- *CT Assessment.* Approximately 8 children per classroom were selected to be assessed in a post-intervention assessment using a stratified random sampling procedure. An equal number of 3- and 4-year-old girls and 3- and 4-year-old boys were randomly selected from each participating classroom. The assessment was used as a measure of CT learning with a view to also pilot-testing the assessment.
- *Post-Study Parent/Caregiver Surveys and Interviews.* To gather information about their experiences (successes and challenges), parents/caregivers completed a survey

Table 3. Demographic Characteristics of Families in Field Study

Characteristic	California		Virginia		Total sample	
	n	%	*n*	%	*n*	%
Ethnicity						
Hispanic or Latino	4	100	2	33.3	6	60
Not Hispanic or Latino	0	0	3	50	3	30
Race						
American Indian or Alaskan Native	0	0	0	0	0	0
Asian	0	0	1	16.7	1	10
Black or African American	0	0	3	50	3	30
Native Hawaiian or Other Pacific Islander	0	0	0	0	0	0
White	3	75	2	33.3	5	50
Highest educational level						
Some high school	0	0	0	0	0	0
High school	2	50	2	33.3	4	40
Some college	2	50	3	50	5	50
College	0	0	1	16.7	1	10

at the end of the 6-week implementation period. The survey included 5 Likert-scale questions specifically pertaining to the intervention:

1. How important/valuable do you think it is to introduce Computational Thinking to your child?
2. Would you be interested in teaching Computational Thinking to your child in the future?
3. On a scale from 1 to 5 (1 means not very helpful and 5 means very helpful), how much do you think the *Better Building* app helped your child learn about Abstraction and Sorting?
4. On a scale from 1 to 5 (1 means not very helpful and 5 means very helpful), how much do you think the *City Walk* app helped your child learn about Algorithms and Math?
5. During the study period, about how often did your child use the program's apps? [Every day, 3-4 times a week, 1-2 times a week]

There were also 2 open-ended response questions:

1. Which of the hands-on activities you tested did you and your child enjoy the most?
2. Which of the hands-on activities you tested did you and your child enjoy the least?

The post-survey was followed by a semi-structured interview to gather deeper feedback and detail on survey responses, and specific feedback on the two digital apps—City Walk and Better Building.

Mixed-Method Analyses and Results

Home visit and family meeting observation notes were used to document implementation successes and challenges of hands-on and digital activities; the degree of children's engagement with the activities; children's and adults' conversations while completing activities; and the scaffolds needed for children to participate successfully in activities (e.g., instructions, feedback, modeling). The purpose of the family meetings was to provide an overview of the CT skill and share the related hands-on activities and digital app games for families to try together on their own or during the meeting. In addition to the family meetings, 9 out of the 10 parents/caregivers responded to the survey and 8 out of the 10 parents/caregivers could be interviewed post-survey. This section presents findings from the survey analyzed quantitatively, and qualitative coding (by 2 researchers) of the home observation notes and post-interview responses to questions pertaining to parents/caregivers' overall experience with the activities, explanations of why an activity was most enjoyable or least enjoyable, and what they thought of the CT resources shared with them. Given the sample size, the coding was done with a view to identifying dominant and recurring themes across the interviews and observations that could provide insights into learner and parent/caregiver experiences with our activity suite. The two researchers jointly reviewed the responses and identified themes described in the section below on why some activities worked (more than others). They then coded the responses independently based on the themes, and then met again to discuss and reach agreement.

Enactment of Home Activities

Based on the review and qualitative analysis of researcher notes during family visits, as well as comments during the interviews, the following feedback emerged on family experiences with each of the activities. Activities such as *Grocery Store Trip* and *Getting Ready for Bed* could not be observed, and the researchers relied on parents/caregivers' descriptions of how those activities went.

Findings from the family visits, surveys, and interviews indicated that, overall, families really enjoyed doing the activities and reported engaging in many of them repeatedly with their children:

"Really good experience. Helps kids with focus and has good directions on what to learn. My daughter is now following directions better. Also really like the Playdough Workshop. She likes to make every item on the cards."

Everything was great. [Child name] and I both like to do a lot of different things, so this was great for us. [Child name] enjoyed everything.

"It is a really great family project. My older kids were helping and playing with [child name]. So it was great to have these games and materials to use at home together."

"[Child name] really loved the activities. She liked to try all of them. She especially loved the Bedtime book and has asked to read it often at night. She also liked making her own book, and is enjoying going back and adding to the book. She is excited to make another book about Getting Ready for School. She did this at school, but would like to make another one at home."

Generally, families appreciated that the activities involved formats familiar to them (e.g., grocery shopping, book reading, cooking, routines, etc). Families also reported that their children liked playing the digital games; all parents/caregivers reported their children used the apps during the study. 78% of parents/caregivers indicated their child using the program's apps at least three times a week. Moreover, all 9 parents/caregivers who completed the post-study survey reported introducing CT to their child as "very valuable" on the survey with nearly all (n=8) expressing highest levels of interest in continuing to teach CT to their child. The following sections provide deeper insights into these findings of family experiences with three of the unplugged activities (one each for our 3 target concepts: Problem decomposition, Algorithms, and Abstraction) and the two digital apps.

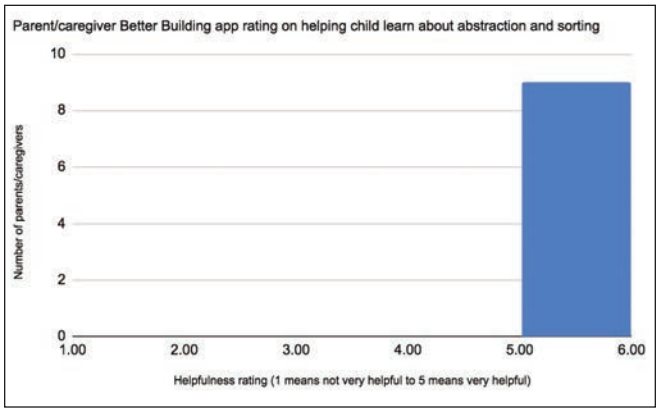
- **Playdough Workshop (Problem decomposition):** The *Playdough Workshop* activity resonated with families,

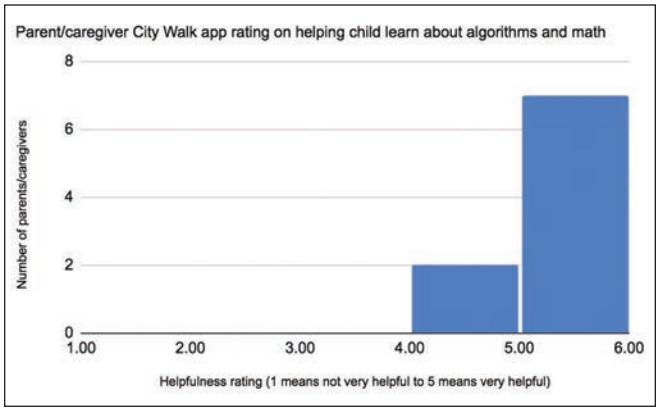
especially given children's familiarity with playdough. Five families named this activity as the one they enjoyed the most on the survey. When prompted to elaborate on this during the interview, a few mentioned the creativity aspect of the activity and noted that the child was excited to share their creations. However, the emphasis appeared to be on working together or discussing the sequence of steps, and the emerging conversations tended to focus on sequencing and math concepts, such as counting and shape identification. The focus on problem decomposition was not always evident. The following quote provides a sense of the kinds of experiences parents shared:

"She really enjoyed working with all of the playdough. She was excited about the different cards with ideas of what to make, and had fun making all of them. She wanted to do each one by herself. Mom went to the dollar store and got some tools so that she could make some shapes easier and use little plastic knives, etc. She was so proud of the things that she created in this activity, and also asks to play with the materials often."

- **Cereal Necklace (Algorithms & Patterns):** Although only one parent/caregiver indicated this activity as their favorite on the survey, two other parents/caregivers explicitly also cited this activity as a favorite in their interview. The families whose children found the activity enjoyable noted in their interviews that their children liked to create patterns; in one instance, the child sorted the cereal prior to making the patterned necklace: "*Told her to choose her favorite cereal. We have the different colored cheerios and the regular cheerios so she sorted the colors. I put it in one big bowl and then she sorted into three smaller bowls. We made a pattern and looped it through the thread.*"

  The parent/caregiver of another child reported how the activity helped her child focus and better follow directions. These observations illustrate the synergy between CT (algorithms) and math (patterns) in this activity and the feasibility of successfully linking CT with math.
- **Grocery Store Trip (Abstraction)** resonated with all families during the initial family visit—they all made some connection with how they might fit the activity into their typical family routines related to grocery shopping. However, when parents/caregivers' shared feedback about this activity after they tried it out or reviewed it in more detail, two families mentioned they were unsure about how best to engage children in the activity, which suggests this may be challenging for some to implement. One parent, who identified this activity as the most challenging in the interview, modified the activity by having the child draw the items to add to their grocery list and then sort into groups. This proved helpful once they were at the grocery store since the child could determine where to find the items on the list: *"This one was difficult because I realized that I go to many grocery stores, for many different items. At first it was hard to think about how to engage [child name], but then she was able to draw the pictures of the items she wanted to add to the list, and then categorize. Then, when shopping she could help figure where to get the items she wanted based on the categories (e.g. strawberries in fruits and goldfish in snacks)."* However, another family remarked on how the activity helped the child, as the child continued to sort unprompted while at the store, noticing the characteristics of the items and sorting into groups by type or color (like in the *Better Building* abstraction app game): *"I feel like it's helped him a lot, especially the one with the grocery store. Now he wants to put it in sets like colors and fruits. He even brings it up without me telling him about. It takes us longer at the grocery store now [because he brings it up]. He goes, "Mommy, let me help you." I was getting vegetables and he wanted me to put in the bag telling me they were green ones. He was sorting the fruits separately in piles on the register."*
- **City Walk App Game (Algorithms & Debugging):** Seven out of nine parents/caregivers rated the *City Walk* digital game a 5 when asked how much they thought *City Walk* helped their child learn about algorithms and math on a scale from 1 (not very helpful) to 5 (very helpful); the other two rated it a 4.





Parents/caregivers expressed that their child found this game enjoyable with some repeatedly returning to it or continuing to play for an extended period. While the game mechanics were generally intuitive for children, three parents/caregivers noted their child needing more extensive support throughout their gameplay. One parent explicitly said their child liked it more than *Better Building* (which was also very well-liked). *"He liked this one more than the building one. I feel like maybe this was a little harder/more challenging. The Better Building was more fast [got through that game more quickly]. He got through it [City Walk], but it took longer."*

The type of scaffolding parents provided varied depending on whether the child needed assistance with either correctly identifying directional arrows or providing directions with multiple directional arrows (visual and spatial thinking). For a couple of parents/caregivers, observing their child's struggles with the game unearthed the concepts (e.g., navigation) they thought they needed to explicitly teach.

- **Better Building App Game (Abstraction & Pattern Recognition):** When asked how much they thought *Better Building* helped their child learn about abstraction and sorting, all parents reported a 5 rating. Based on parent/caregivers' feedback during the interview and researcher observations of children's gameplay, children enjoyed the game and easily played the game independently. For example, *"First time, he ever did that app, he was able to grasp that sorting really well. With his toys, he puts them back and puts them into the categories we did at home. Better organization for when they need to find something. He really enjoyed the sorting. He liked getting to build his pattern. He was engaged in this one a little more."* Children's successful completion of levels and the extent to which they needed additional support seemingly corresponded with their familiarity of the characteristics of objects being sorted within the game (e.g., shape, color, and size). One parent/caregiver initially thought the game would be easy for her child given the child's familiarity with shapes and color yet found the game appropriately challenging as the child needed scaffolding at times around the characteristic the object was being sorted by. Another parent/caregiver highlighted the benefit of the categories and/or labels within the game, noting this similarity to the *Grocery Store Trip* activity (which is also an abstraction-focused activity). To increase game complexity, some parents/caregivers suggested including more difficult levels or involving less familiar shapes: *"That one was also pretty easy for her to do so adding difficulty levels (different colors and shapes). She goes more to the City Walk."*

Similar to the hands-on activities, there were opportunities across both digital games for children to jointly engage in gameplay with siblings or family members. In addition, for one family in particular, the child's gameplay sparked math related conversations around counting/cardinality and shape identification. There were also a few instances where we observed parents/caregivers taking an active role in mediating their child's gameplay by either providing "in the moment" scaffolding or posing questions to their child.

Why Certain Activities Worked (More or Less Than Others)

The survey asked parents/caregivers to specify which activity (or activities) their child enjoyed the most and least. For most enjoyable, the *Playdough Workshop* activity ranked the highest (5 out of 9 parents/caregivers); 3 out of 9 marked the *Playing with Dice* activity; and *Cereal Necklace* and *Getting Ready for Bed* received 1 vote each with one family reporting two favorite activities. These numbers changed slightly during the interviews with 3 parents/caregivers mentioning *Cereal Necklace* as the most enjoyable. Parents/caregivers were probed during the interview of their overall impressions of the activity suite, why a particular activity was their child's favorite (or least favorite), and their specific feedback on *City Walk* and *Better Building* digital apps. It is interesting to note that many parents/caregivers spoke of their children continuing to do the activity, suggesting that that was an indicator by which the activity was judged. We coded the responses for themes as the insights are valuable in future iterations and activity designs and help *shine a light especially on those features of activity designs unique to home and family involvement.*

The following key themes emerged from coding parents/caregivers' explanations of why a particular activity was most enjoyable. We have added a few parent/caregiver quotes that embody the sentiment:

- **Familiarity with materials** (playdough or cereal) or activity theme (such as shapes) and/or **alignment with usual home routine** (how they did grocery shopping or got ready for bed).

  [Playdough Workshop] "First of all, my kids in general already like playdough. For them, it wasn't like "Oh, we're going to learn." It was something that we're able to do and they still continue to do."
- **Level of difficulty.** Too much challenge (as in Grocery Store Trip) or too little challenge (as in Better Building) was a problem for promoting engagement.

  [Grocery Store Trip] "Grocery list was too hard. She did not understand what to do."

  [Better Building] "I think only sorting 2 colors has gotten too easy for her. Including more shapes to make it more difficult."
- **Levels of support needed.** While age-appropriateness was a consideration in our activity designs throughout, some activities were less engaging likely because ideas

therein required high levels of scaffolding for the child as in *Robot in the City* (as well as directional arrows in *City Walk* for a few children).

[City Walk] "She also had a hard time with the different directions. She may just need more practice, but for now it seems a bit advanced."

- **Creativity/creating something (as opposed to enacting).** The two most popular activities—*Playdough Workshop* and *Cereal Necklace* involved children creating physical artifacts. Many parents/caregivers expressed creativity or creating things as a positive feature when explaining why these activities were enjoyable.

  [Cereal Necklace] "She was so proud of the things that she created in this activity, and also asks to play with the materials often."

  [Playdough Workshop] "I feel like it worked well because it was more sharing who was gonna do what, specifying who's going to do this. Also used him to use his creativity. He asks me 'Can I use this color?' and he uses the color he wants."
- **Family / siblings joining in the activity.** Parents/caregivers often mentioned siblings playing with the activity or joining in to engage jointly with the activity. Activities that fostered this were seen as more enjoyable, most notably, *Playdough Workshop*, *Playing with Dice*, and *Cereal Necklace*.

  [Playdough Workshop] "It's helped not only him, but also my younger daughter. Telling each other who is going to do what. I really like that one."

  [Playing with Dice]"This one my older children really liked playing together. It got them all moving and they had fun."

  [Digital Apps] "I feel like in my opinion both activities, you know how I said before he's second year in preschool, it's like a practice he continues to do. He just turned 5 and my son teaches my daughter. She's adapting to what he teaches."

In addition, other themes worth noting were:

- **The home-school connection was reinforced in some activities.** For example, children were excited about making the *Getting Ready for Bed* book because of the *Getting Ready for School* book they made in school. *Better Building* was enjoyable for some children because of its familiarity having been introduced to it in class.
- **Parents were very appreciative of the CT training and resources provided to them.** None was familiar with the term "Computational Thinking" prior to the intervention, and they all appreciated the video and the connections the examples made with activities familiar to them. They welcomed the partnership with their child's school and were also keen for other parents/caregivers and families to experience it.

"Video is a good way to walkthrough; can repeat it whenever you need a refresher."

"I know you told me the definitions, but seeing the examples helped me understand it more. These activities can show parents what children know and don't know, be involved with them. It gave me an idea of what my son should practice."

"The activities really help children and parents engage together at home. And it is really great for the Head Start center to have this partnership. Really hope that the program continues and more children/families get an opportunity to participate!"

"I'm not sure if there's any way to make it more available [to more parents]."

CT Learning

The assessment data gathered from the CT assessment designed as part of the project was used to examine pre-post changes in student learning. A summated score was used to exploratorily examine the promise of the resources developed; the estimated reliability of the summated score was 0.72 at pre-test and 0.78 at post-test. The distribution of the data was examined to ensure there were no outliers and a simple regression model with two predictors only (pre-test score and condition) was conducted to examine children's change in post-test scores. Significant improvements were detected from pre to post for children *who participated in the home+school* connection condition, relative to the comparison classrooms *($t = -3.056$ (45), $p < .005$)*. Interestingly, *a significant effect was not detected for children who participated in the classroom only condition, relative to the comparison condition*. These findings (albeit with a small sample) when combined with the overwhelmingly positive feedback from families makes this a salient result, given that involving families was a unique element of this design research.

Discussion

Our research presents family experiences with integrating CT activities into their homes as part of a broader effort to integrate CT into children's early STEM learning in school and home. Our qualitative research analyzing the home component provides thick data and rich insights into family experience and provides many learnings and ideas for the broader field and educational goal for strengthening early STEM learning. The benefit of involving the family was apparent in parents'/caregivers' comments that clearly indicated how they enjoyed it as a "family project". Our findings highlight the promise of introducing early CT to support early learning, and especially involving families in the process. Results from our research also identified challenges that should be addressed in future iterations of this design research. Parents/caregivers were enthusiastic about the intervention and they appreciated being

introduced to the new idea of "Computational Thinking." High learner engagement levels in homes underscore the value of co-designing activities with parents/caregivers (and teachers) that naturally fit the ecology of preschool homes (and classrooms). Our findings on successful engagement with abstraction and debugging are resonant with recent findings (e.g., Yadav et al., 2019) that indicate that while young children may be able to engage in coding and algorithmic thinking, other CT skills may be productive entry points for young learners and more naturally aligned with the hands-on activities young children experience at home and school.

The overwhelmingly positive response from parents/caregivers and the significant pre-to-post finding in the home+school condition underscores the importance of efforts that link home and school to support preschool children's STEM learning. Valuing learners' ways of talking, thinking, and interacting in schools that are consonant with the practices that they bring from home is an equity-centered pedagogical practice (Nasir et al., 2020) that can enrich STEM and CT learning in elementary and secondary levels too. Our approach also helps families from economically disadvantaged communities be better prepared to support and strengthen their child's out-of-school STEM learning.

While some reasons for the overall success of the activities– such as, alignment with home routine, familiarity with materials, involvement of parents/caregivers and siblings, appropriate levels of difficulty and scaffolding– were intuitive and to be expected, the following findings were interesting and merit attention. (1) Algorithmic thinking in K-12 classrooms is often fostered through navigational activities—in physical or virtual space. This has perhaps been influenced by the ideas of turtle geometry that motivated Papert (1980)'s work with Logo and children's programming. However, amongst our unplugged activities, the navigational activity (*Robot in the City*) was not very popular with families when compared with *Cereal Necklace* and *Playing with Dice*, which involved creating artifacts or engaging in movement in fun and engaging ways. The overwhelming success of *Playdough Workshop* also underscored the value of activities that involve creativity and creating artifacts. Perhaps it was because the navigational directions and instructions in *Robot in the City* were harder to implement and needed more scaffolding. This suggests that educators and designers should expand the repertoire of algorithmic activities beyond navigational ones to include those that involve creating artifacts, and preferably with materials familiar to young children. (2) It was interesting that though unplugged activities often provide a scaffold for digital interactions (in the context of programming, e.g. Grover, Jackiw, & Lundh, 2019), *Robot in the City* (which was meant to be the unplugged activity to scaffold the *City Walk* digital app) seemed harder for preschool children than *City Walk*; and elements of *City Walk* - such as directions (left/right)– came in handy for children as they played *Robot in the City*. Our data suggest that this was because *Robot in the City* required the adult to situate the materials and explicitly facilitate/scaffold the activity. This distinction in experience between digital and unplugged activities covering the same concept highlights what kind of activities can leverage and benefit from digital affordances (as *City Walk* does).

Lastly, holding children's attention is important and some activities were clearly better on this criterion than others. However, we found that unexpected factors can play a role in addition to the level of challenge; sometimes other distractions come in the way of implementation especially with an audience of preschool learners. For example, other happenings around the library (where the family meeting was taking place with the researchers) would draw the child away; or a robot figurine in *Robot in the City* would distract the child from focusing on the solution to the navigational problem. This is perhaps a relevant takeaway for children in the lower primary age group as well.

Implications

Our project helps contribute much-needed evidence to the research base on early CT and inform future development of evidence-based home and school resources for children to learn CT, mathematics and science through integrated activities and helps surface both successful approaches as well as challenges. The prototype activities we co-designed, developed, and empirically investigated help improve our understanding of productive integration points, activity formats familiar to preschoolers that support the integration of CT and STEM, as well as how digital tools complement and strengthen the learning resulting from hands-on activities in early childhood. It is important to note that our work focused on a subset of CT skills and that future research is needed to explore how a wider set of CT skills can be meaningfully promoted in early learning.

Combining family involvement with school-based activities was a uniquely successful element of this research that has lessons for CT learning at elementary and other K-12 levels as well. It underscores the benefits of culturally responsive and relevant teaching that involves families and communities and gives due credence to learners' cultures and practices in the learning process. Findings from our study also highlighted that families welcomed and valued support for them to promote early CT at home—resources to help parents and caregivers understand what CT is and recognize when they naturally engage in CT in their everyday lives was particularly helpful to them. Parents and caregivers noted that visual and engaging formats are especially helpful and appreciated the introductory video co-designed as part of the project. It is worth noting that meetings with parents/caregivers had to be scheduled around their availability and this sometimes happened

outside the home—at a local library or playground. Such flexibility is required when working with parents/caregivers, but it sometimes introduces additional distractions and attendant challenges in enacting activities with the little ones. Nevertheless, this project reinforces earlier findings on the positive impact of family involvement on children's early STEM learning. We encourage educators and designers to engage families in the design and use of CT and computing-related activities that will fit into the cultural milieu of learners.

We experimented with known as well as hitherto unexplored operationalizations of CT for early learners with new learnings from each. For example, our use of arrows for navigational activities is inspired by other tools (such as LightBot Jr. and Scratch Jr.). Our findings of young learners' struggle with directionality is one with broad implications. Conversely, our success in activities that involve algorithmic thinking through creating artifacts was promising and merits attention. Our operationalization of abstraction as observing key characteristics of classes of shapes for making block building activities serves as an exemplar (as it also potentially connects to future CT and programming through helping learners develop valuable foundational skills of functional abstraction related to objects and classes in object-oriented programming). Future research efforts will continue to explore broader design principles that can inform the development of both activities with tangible materials and digital activities to inform the design of additional CT resources for teachers and families of preschool learners.

References

Clements & Sarama (2014). Learning trajectories in mathematics education. Mathematical thinking and learning, 6(2), 81-89.

Dong, Y., Catete, V., Jocius, R., Lytle, N., Barnes, T., Albert, J., ... & Andrews, A. (2019, February). PRADA: A practical model for integrating computational thinking in K-12 education. In Proceedings of the 50th ACM SIGCSE (pp. 906-912).

Duncan, G. J., Dowsett, C. J., Claessens, A., Magnuson, K., Huston, A. C., Klebanov, P., ... & Sexton, H. (2007). School readiness and later achievement. Developmental psychology, 43(6), 1428.

Gelman, R., & Brenneman, K. (2004). Science learning pathways for young children. Early Childhood Research Quarterly, 19(1), 150-158.

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. (2018). Computational Thinking: A competency whose time has come. In S. Sentance, E. Barendsen, & S. Carsten (Eds.), *Computer Science Education: Perspectives on Teaching and Learning in School*. pp. 19–38. Bloomsbury.

Grover, S., Jackiw, N., & Lundh, P. (2019). Concepts before coding: non-programming interactives to advance learning of introductory programming concepts in middle school. *Computer Science Education, 29*(2-3), 106-135.

Horn, M. S., AlSulaiman, S., & Koh, J. (2013, June). Translating Roberto to Omar: computational literacy, stickerbooks, and cultural forms. In *Proceedings of the 12th International Conference on Interaction Design and Children* (pp. 120-127). ACM.

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.

Leibham , M. B., Alexander, J. M., & Johnson, K. E., (2013). Science interests in preschool boys and girls: Relations to later self-concept and science achievement. Science Education, 97, 574-593.

Ma, X., Shen, J., Krenn, H. Y., Hu, S., & Yuan, J. (2016). A meta-analysis of the relationship between learning outcomes and parental involvement during early childhood education and early elementary education. *Educational Psychology Review, 28*(4), 771-801.

Mittermeir, R. T. (2013). Algorithmics for preschoolers: A contradiction? *Creative Education, 4*(9), 557.

Moll, L. C. (2015). Tapping into the "hidden" home and community resources of students. Kappa Delta Pi Record, 51(3), 114-117.

Moll, L. C., Amanti, C., Neff, D., & Gonzalez, N. (1992). Funds of knowledge for teaching: Using a qualitative approach to connect homes and classrooms. Theory into practice, 31(2), 132-141.

Nasir, N., Lee, C., Pea, R., & De Royston, M.M. (2020). (Eds.), Reconceptualizing learning: a critical task for knowledge-building and teaching. Routledge Handbook of the Cultural Foundations of Learning. New York: Routledge.

Penuel, W. R., Roschelle, J., & Shechtman, N. (2007). Designing formative assessment software with teachers: An analysis of the co-design process. *Research and practice in technology enhanced learning, 2*(01), 51-74.

Pinkard, N., Martin, C. K., & Erete, S. (2020). Equitable approaches: opportunities for computational thinking with emphasis on creative production and connections to community. *Interactive Learning Environments, 28*(3), 347-361

Presser, A. L., Dominguez, X., Goldstein, M., Vidiksis, R., & Kamdar, D. (2019). Ramp It UP!. Science and Children, 56(7), 30-37.

Roshan, P. K., Jacobs, M., Dye, M., & DiSalvo, B. J. (2014, November). Exploring How Parents in Economically Depressed Communities Access Learning Resources. In *GROUP* (pp. 131-141).

Sandoval, W. A., & Bell, P. (2004). Design-based research methods for studying learning in context: Introduction. Educational psychologist, 39(4), 199-201.

Schweingruber, H. A., Duschl, R. A., & Shouse, A. W. (Eds.). (2007). Taking Science to School: Learning and Teaching Science in Grades K-8. National Academies Press.

Scott, K. A., Sheridan, K. M., & Clark, K. (2015). Culturally responsive computing: A theory revisited. *Learning, Media and Technology, 40*(4), 412-436.

U.S. Department of Health and Human Services. (DHHS; 2015). Head Start Early Learning Outcomes Framework. https://eclkc.ohs.acf.hhs.gov/sites/default/files/pdf/elof-ohs-framework.pdf

Vahey, P. J., Reider, D., Orr, J., Lewis Presser, A., & Dominguez, X. (2018). The Evidence Based Curriculum Design Framework: Leveraging Diverse Perspectives in the Design Process. International Journal of Designs for Learning, 9(1), 135–148. https://doi.org/10.14434/ijdl.v9i1.23080

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.

White House, Office of the Press Secretary. (2016, January 30). Computer science for all. https://www.whitehouse.gov/blog/2016/01/30/computer-science-all

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

Yadav, A., Larimore, R., Rich, K., Schwarz, C. (2019). Integrating computational thinking in elementary classrooms: Introducing a toolkit to support teachers. In *Proceedings of SITE International Conference 2019*. Chesapeake, VA: AACE.

AUTHOR BIOGRAPHIES

Bataul Alkhateeb is a Ph.D. candidate in the School of Education at the University of Delaware. She received a bachelor's degree in Psychology from the University of Baltimore and a master's degree in Curriculum, Instruction, and Assessment from Qatar University. While pursuing her master's degree, she grew interested in teacher preparation programs and improving curricula for training teachers in computer science and technology literacy. Her thesis assessed the educational technology self-efficacy beliefs of preservice teachers. Currently she examines pre-service teacher learning of computational thinking through practice-based simulations as well as equity issues in computer science professional development for in-service teachers.

Mahtob Aqazade is a postdoctoral research associate at Rice University School Mathematics Project. Her research focuses on elementary mathematics education, particularly on the role of conflict induction and conflict resolution in learning of conceptually challenging mathematical concepts using stories and contexts. In her work with elementary teachers, she is interested in exploring teachers' learning processes when interpreting and incorporating new instructional strategies and practices. Mahtob has a Ph.D. and M.S. in mathematics education from Purdue University and B.S. in Industrial Mathematics.

Bianca Bennett is the Global Studies Manager for DC Public Schools, managing the school systems global education enrichment programming and curriculum development. Bianca's career in education includes teaching, coaching and professional development, program development and management, and curriculum design. Bianca's education focus and passions include ensuring equity and access to advanced academic and educational enrichment experiences for underrepresented, culturally and linguistically diverse students. Bianca has a Masters in Early Childhood Education from Georgia State University and a Masters in Public Administration from City University of New York, Baruch College.

Marina Umaschi Bers is a professor at the Eliot-Pearson Department of Child Study and Human Development with a secondary appointment in Computer Science Department, at Tufts University. She heads the interdisciplinary DevTech research group. Her research involves the design and study of innovative learning technologies to promote young children's positive development. Dr Bers is the co-creator of the free ScratchJr programming language and the creator of the KIBO robotic kit. She is the author of five books on the topic of education, new technologies and children. Her newest book is called "Beyond Coding: How Children Learn Human Values through Programming". Marina has an MS and PhD from the MIT Media Lab, where she worked with Seymour Papert. Her undergraduate degree is from Buenos Aires University.

Laura Bofferding is an associate professor in Curriculum and Instruction at Purdue University. Her research focuses on mathematics education, particularly on conceptual change and young children's understanding of negative numbers. She also investigates the impact of worked examples, contrasting cases, and instructional sequences on supporting students' thinking. Recently, she has expanded her research into the areas of spatial and computational thinking, investigating the role of using worked examples to help young students learn to program. Laura has a Ph.D. in Curriculum Studies and Teacher Education and an MA in Learning, Design, and Technology from Stanford University. Her undergraduate degree is in elementary education.

AUTHOR BIOGRAPHIES

Lautaro Cabrera is a postdoctoral researcher at the University of Maryland, College Park. His research focuses on the integration of computing into Elementary schools, particularly through teacher education and professional development. More broadly, his work lies at the intersection of technology, psychology, and learning, with special attention to how students and teachers develop conceptualizations of ideas and how tools support that development. He has an undergraduate degree in Psychology from Ohio Wesleyan University and a Ph.D. in Technology, Learning, and Leadership from the University of Maryland, College Park.

Lizhen Chen is a postdoctoral researcher at Ohio University. Her research focuses on preservice and in-service teachers' classroom discourse and gestures, particularly on their probing practices. She also investigates elementary students' mathematics learning and the caring relationships between teacher educators and preservice teachers. Lizhen has a Ph.D. in mathematics education and an MA in English linguistics.

Merijke Coenraad is a PhD Candidate in the Department of Teaching & Learning, Policy & Leadership in the College of Education at the University of Maryland. Her research focuses on the intersections of educational technology and equity including the creation of materials, platforms, and experiences in partnership with teachers and youth through participatory design methods. Merijke has an M.Ed in Curriculum and Instruction from Boston College and a B.S. in Elementary Education and Spanish and Hispanic Studies from Creighton University.

Ximena Dominguez is the director of early STEM research at Digital Promise. Her research focuses on young children's STEM learning across home and school and involves partnerships with public preschool educators, curriculum developers, media designers and families from historically underserved communities to co-design equitable learning experiences for young children. In addition to studying how science and mathematics can be promoted early in childhood, her current work investigates how engineering and computational thinking can be introduced to support play and early learning and explores how STEM domains can be feasibly and meaningfully integrated in preschool classrooms. Her work also involves developing resources for multilingual learners and explores the affordances of technology and media for supporting early STEM teaching and learning. Ximena earned an M.S.Ed. in education from the University of Pennsylvania and a Ph.D. in applied developmental psychology from the University of Miami.

Andrew Elby, a professor of education at the University of Maryland, College Park, normally focuses on science teaching and learning but has recently been exploring the integration of computational thinking into science and integrated STEM learning. His research often focuses on teachers' and students' epistemological views about what counts as learning in the classroom. His graduate degrees from University of California, Berkeley, are in Physics, Philosophy of Physics, and Mathematics & Science Education.

Janet Bih Fofang is a doctoral student in the Technology, Learning, and Leadership program at the University of Maryland. She has a strong interest in the learning sciences and technology design. Her current research is primarily concerned with robotics and computational thinking in K-12 education, designing and implementing tools and environments that can support the integration of CT in STEM contexts using robots. Prior to coming to the University of Maryland, Janet taught electronics in high schools and technical vocational colleges in Cameroon. Janet holds a Masters degree in electrical engineering from the university of Douala - Cameroon.

Madhu Govind is a doctoral candidate in the Eliot-Pearson Department of Child Study and Human Development at Tufts University and a graduate research assistant at the DevTech Research Group. Her interests broadly encompass the ways in which innovative education technologies and pedagogies can promote children's learning and creative expression. Her current work focuses on PK-2 teachers' attitudes and experiences with coding and robotics education.

Sara Gracely is a Project Manager at SRI International. She is engaged in multiple research projects in early childhood STEM education and social-emotional learning. Sara also leads SRI Education's Student Behavior Blog which provides the latest information about evidence-based approaches to support all students' positive behavior, mental health, and well-being. She holds a B.A. in Psychology with an emphasis in early childhood development from the University of Wisconsin-Madison.

Shuchi Grover, is a computer scientist and learning scientist by training. She has been immersed in PK-12 computer science education in formal and informal settings for over two decades. Formerly a senior researcher at SRI International's Center for Technology in Learning and Visiting Scholar at Stanford University, she is currently senior research scientist at Looking Glass Ventures where she leads several NSF-funded projects involving research & design of curriculum, assessments, tools, and environments that help develop 21st century competencies in topics such a computing, STEM+CS integration, AI, and cybersecurity as well as issues of neuro-diversity, gender equity, and teacher preparation. Shuchi has a Ph.D. in Learning Sciences & Technology Design with a focus on K-12 CS Education (Stanford University), master's degrees in education (Harvard University) and computer science (CWRU, Cleveland), and bachelor's degrees in computer science and physics (BITS Pilani, India).

Ana-Maria Haiduc. Building on her eight years of experience as a mathematics teacher, Ana-Maria's research uses caring theory to investigate teachers' interactions with curriculum resources. Ana-Maria works to understand teachers' curricular decision-making, specifically as they relate to differentiated instruction. She supports efforts to use incorrectly worked examples in instruction and develop an equitable curriculum. Additionally, her interests focus on measurement, coordinate geometry, and students' thinking. Ana-Maria holds a master's degree in mathematics from Purdue University Northwest and is currently a Ph.D. graduate student in Mathematics Education at Purdue University West Lafayette.

AUTHOR BIOGRAPHIES

Sharin Jacob is a PhD in Education candidate at the University of California, Irvine. She has worked on designing and evaluating the Elementary Computing for All curriculum. This curriculum integrates computational thinking (CT) with English Language Arts content to simultaneously develop multilingual students' CT, language, and literacy skills. Most recently, Sharin has been recognized as a UCI Public Impact Distinguished Fellow for her commitment to bringing actionable change for multilingual students in computing.

Madison Kantzer is the Manager of Assessment and Instruction Innovation at District of Columbia Public Schools. She has 10 years of experience in experiential learning and inquiry based education. Madison has worked with the Sphero.Math project since February 2020. Madison has a Masters in Social Studies Teaching and Learning from Columbia University, Teachers College,

Danae Kamdar is an Early STEM Education Researcher at Digital Promise. She is engaged in multiple research and development projects that aim to better understand how to promote STEM teaching and learning in both early childhood classroom settings and homes. Danae's work focuses on building partnerships with teachers and families to co-design early learning resources that connect to children and families' everyday lives. She also engages in the development of child assessments that are intended to examine what young children are capable of learning and what resources and instruction may lead to increased learning – with an ultimate goal of translating research findings about early STEM education into innovative, equitable resources that impact children's learning and teachers' practices. She holds an M.S. in Early Childhood Education from New York University. Prior to becoming a researcher, she was a teacher and classroom coach in diverse early learning classrooms, and this experience informs her work each and every day.

Diane Jass Ketelhut is Professor of Science, Technology and Math Education at the University of Maryland. Her research centers on improving self-efficacy, learning and engagement in computational thinking and science for students and teachers, particularly through scientific inquiry experiences within virtual environments. She is currently the Principal Investigator on two NSF-funded projects: *CT-->PSTE* that investigates the integration of computational thinking into elementary preservice, undergraduate science teacher education; and the ACT project, which explores best practices for helping teachers provide culturally relevant experiences for all elementary children to participate in and engage with computational thinking integrated into their science lessons. Certified in secondary science, she was a science/math teacher for Grades 5–12 for 12 years. Diane received a B.S. in Bio-Medical Sciences from Brown University, and her doctorate in Learning and Teaching from Harvard University.

Heather Killen is a PhD Candidate in the College of Education at the University of Maryland, College Park. Heather researches adult STEM learning in informal and formal contexts, including within teacher professional development experiences. She explores how fostering agency, cooperative learning, and the integration of computational thinking practices can support virtual and physical learning communities. Heather has a master's degree in biology from Boston University's Marine Program in Woods Hole, MA and experience as an ecologist and resident biology faculty for community college. Her undergraduate degree is in biology.

Sezai Kocabas is a Ph.D. student in Curriculum and Instruction at Purdue University. His research focuses on young children's computational and mathematical thinking, particularly forms of worked examples and the role of analyzing worked examples to learn programming concepts. Recently, he started investigating young children's spatial and computational thinking during block composition and decomposition tasks. Sezai has an MS in Teaching, Learning and Culture from Texas A&M University. His undergraduate degree is in elementary education.

Theodore J. (TJ) Kopcha is an Associate Professor of Learning, Design, and Technology at the University of Georgia; he holds a B.S. in Mathematics Education and an M.A. in Curriculum and Instruction from the University of Connecticut. As a former secondary mathematics teacher, TJ's research explores how innovative technologies like educational robots can support children's embodied understanding of mathematics. His research, which has been funded by UGA's Public Service and Outreach, the National Endowment for the Humanities, and the Spencer Foundation, provides support to rural communities who are integrating technology through approaches like project-based learning.

Tiffany Leones is an early STEM education researcher at Digital Promise. She works on projects engaging in participatory research and co-design to develop equitable early STEM innovations across school and home. Her research interests more broadly include promoting family engagement and exploring the role of educational media and technology to support early learning. Tiffany has an M.Ed in Educational Psychology – Applied Developmental Science from the University of Virginia and a B.A. in Psychology with a minor in Education and Applied Psychology from the University of California, Santa Barbara.

Peter Moon is a doctoral student in the Center for Math Education at the University of Maryland. A former high school math & programming teacher, he's interested in how students express STEM understanding in multimodal ways and how computational thinking can inspire greater mathematical understanding. Peter has an M.A.T. in Secondary Mathematics from Loyola University Maryland and a B.A. in Psychology from the University of Pennsylvania.

Chrystalla Mouza is Distinguished Professor and Director of the School of Education at the University of Delaware. Her research focuses on teacher learning and professional development in emerging technologies and computer science education. She directed several projects aimed at improving teaching and learning with technology in high-need schools and preparing in-service and pre-service teachers to integrate computational thinking with content area curricula. Dr. Mouza is the recipient of the 2010 Distinguished Research in Teacher Education Award from the Association of Teacher Educators and current Editor of the journal of Contemporary Issues in Technology and Teacher Education. Dr. Mouza holds Ed.D., M.Ed. and M.A. degrees in Instructional Technology and Media from Teachers College, Columbia University.

AUTHOR BIOGRAPHIES

Anne Ottenbreit-Leftwich is the Barbara B. Jacobs Chair in Education and Technology. She is a Professor and Interim Chair of Instructional Systems Technology within the School of Education and an Adjunct Professor of Computer Science at Indiana University – Bloomington. Dr. Leftwich's expertise lies in the areas of the design of K-12 curriculum resources (particularly focused on technology and computer science), the use of technology to support teacher education, and development/implementation of professional development for teachers and teacher educators. Dr. Leftwich investigates ways to teach K-12 computer science, as well as ways to prepare preservice and inservice teachers to teach CS. She is a co-PI for the ECEP alliance, which seeks to broaden participation in computing at the K-16 levels. She is also a co-founder of CSforIN, which focuses on increasing CS access opportunities for all K-12 Indiana students. Her research focuses on the adoption and implementation of technology and computer science at the K-12 levels, particularly at the elementary level.

Miranda Parker is a Postdoctoral Scholar at the University of California, Irvine. Her research is in computer science education, where she focuses on topics of assessment, achievement, and access. Miranda received her B.S. in Computer Science from Harvey Mudd College and her Ph.D. in Human-Centered Computing from the Georgia Institute of Technology, advised by Mark Guzdial.

Jan Plane, a principal lecturer of computer science and the director of the Iribe Initiative for Inclusion and Diversity in Computing (I4C) at the University of Maryland, College Park, has graduate degrees in both computer science and education; therefore, she focuses on computer science curriculum, pedagogical methods and underrepresented populations in computing. For 15 years, she worked extensively in sub-Saharan Africa and Afghanistan developing and supporting computer science education programs for universities there. Now, she is the founding director of both the Iribe Initiative for Inclusion and Diversity in Computing (I4C) and the Maryland Center for Women in Computing (MCWIC) and is involved with projects locally encouraging both quality of content and access for underrepresented populations. Jan has graduate degrees from the University of Wisconsin, Milwaukee (computer science) and the University of Maryland, College Park (education).

Emily Relkin is a Ph.D. student at the Eliot Pearson Department of Child Study and Human Development at Tufts University and is a graduate research assistant at the DevTech Research Group. Her research focuses on understanding and assessing the development of computational thinking skills in young children. She developed and validated TechCheck, a novel unplugged computational thinking assessment for early childhood that is being used in research and educational settings around the world.

Scott Sheridan is a Ph.D. candidate in the School of Education at the University of Delaware. He received a bachelor's degree in German Language from Bates University and a Masters degree in Educational Technology from the University of Connecticut. Prior to joining the doctoral program at the University of Delaware, Scott worked as a secondary school teacher, coach, and administrator. His research interests focus on teacher preparation and professional development in computer science education. His dissertation examines the ways in which elementary and middle school teachers apply computer science related content, pedagogy, and technology in their classrooms following their participation in professional development.

Phil Vahey is the Director of Applied Learning Sciences at Houghton Mifflin Harcourt. Prior to this, including during the research reported on here, he was the Director of Strategic Research and Innovation in the Education Division of SRI International. At SRI he researched the design, development, and evaluation of technologies and learning activities that enhance learning of conceptually difficult STEM topics, leading several NSF- and Department of Education-funded studies. Phil received his PhD and MA in Education from the University of California, Berkeley, and his undergraduate degree in Math and Computer Science from McGill University.

Janet Walkoe is an Associate Professor in the Center for Mathematics Education (CfME) at the University of Maryland. She earned her Doctorate from Northwestern University in the Learning Sciences in 2013. She also holds an MS in Mathematics from the University of Illinois at Chicago and a BA in Mathematics from the University of Chicago. Before enrolling in graduate school, Janet taught secondary mathematics for ten years and earned National Board Certification in 2003. Janet is the PI on an NSF funded grant: CAREER: Exploring Teacher Noticing of Students' Multimodal Algebraic Thinking, where she investigates how children's lived experiences can provide resources for teachers to leverage in formal algebraic instruction. In particular, she is interested in how teachers attend to and make sense of student thinking and other student resources including but not limited to student dispositions and students' ways of communicating mathematics.

Margaret Walton is a PhD Candidate in the Department of Teaching & Learning, Policy & Leadership in the College of Education at the University of Maryland. Her research interests focus on teachers' attention to student thinking and how teachers make sense of student thinking. She is particularly interested in how math teacher educators support teachers in learning to center student ideas during instruction. Prior to studying at Maryland, Margaret was a high school math teacher in Washington, D.C. Margaret has a B.A. in Economics from Boston College and a Master's in Teaching from the University of Virginia.

Mark Warschauer is Professor of Education at the University of California, Irvine, where he directs both the Digital Learning Lab and the Elementary Computing for All Project. His research focuses on the design, implementation, and evaluation of digital environments that promote language and literacy development and STEM learning among diverse learners, especially multilingual students. He is a member of the National Academy of Education.

David Weintrop is an Assistant Professor in the Department of Teaching & Learning, Policy & Leadership in the College of Education with a joint appointment in the College of Information Studies at the University of Maryland. His research focuses on the design, implementation, and evaluation of effective, engaging, and equitable computational learning experiences. His work lies at the intersection of design, computer science education, and the learning sciences. David has a Ph.D. in the Learning Sciences from Northwestern University and a B.S. in Computer Science from the University of Michigan.

AUTHOR BIOGRAPHIES

Cheryl Wilson is a doctoral student in the Learning, Design, and Technology program at the University of Georgia. A former programmer at an institute of higher education, she's interested in exploring the cross-section of maker education and learner STEM identity as a means to broaden participation in STEM. Cheryl has an M. Ed. in Learning, Design, and Technology from the University of Georgia and a B. S. in Information Technology from Georgia Southern University.

Aman Yadav is a Professor of Educational Psychology and Educational Technology in the College of Education at Michigan State University with extensive experience in research, evaluation, and teacher professional development. His areas of expertise include computer science education, problem-based learning, and online learning. His research and teaching focus on improving student experiences and outcomes in computer science and engineering at the K-16 level. His recently co-edited book, *Computational Thinking in Education: A Pedagogical Perspective* tackles how to integrate computational thinking, coding, and subject matter in relevant and meaningful ways. His work has been published in several leading journals, including *ACM Transactions on Computing Education*, *Journal of Research in Science Teaching, Journal of Engineering Education*, and *Communications of the ACM*.

Dayae Yang is a doctoral student in the Learning, Design, and Technology program at the University of Georgia. She is interested in how students engage in computational thinking using different learning tools including educational robotics and games. Dayae has an M.A. in Instructional Systems Technology from Indiana University.

Hui Yang is an education researcher (STEM & CS) at SRI International . Her research interests focus on the design of technology-rich learning environments that engage learners into meaningful experiences and teacher preparation in computer science education. Prior to joining SRI, Dr. Yang was a postdoctoral associate in the department of Information Science at Cornell University. Dr. Yang completed her Ph.D. at the University of Delaware.

A Special Research Publication

Association for
Computing Machinery