Interactive Learning Environments

ISSN: 1049-4820 (Print) 1744-5191 (Online) Journal homepage: https://www.tandfonline.com/loi/nile20

From classroom lessons to exploratory learning progressions: mathematics + computational thinking

Maya Israel & Todd Lash

To cite this article: Maya Israel & Todd Lash (2020) From classroom lessons to exploratory learning progressions: mathematics + computational thinking, Interactive Learning Environments, 28:3, 362-382, DOI: 10.1080/10494820.2019.1674879

To link to this article: https://doi.org/10.1080/10494820.2019.1674879

Published online: 21 Oct 2019.

INTERACTIVE LEARNING ENVIRONMENTS
2020, VOL. 28, NO. 3, 362–382
https://doi.org/10.1080/10494820.2019.1674879

Routledge
Taylor & Francis Group

From classroom lessons to exploratory learning progressions: mathematics + computational thinking

Maya Israel [a] and Todd Lash[b]

[a]Department of Educational Technology, School of Teaching and Learning, University of Florida, Gainesville, FL, USA; [b]Department of Special Education, University of Illinois, Champaign, IL, USA

ABSTRACT

This paper presents findings from a two-year qualitative study examining integration of computer science (CS) and computational thinking (CT) into elementary mathematics instruction. Integrated units were developed by elementary teachers and CS/CT coaches with support from university faculty with expertise in CS/CT and elementary mathematics. CS/CT instruction primarily relied on the Scratch environment, although some lessons made use of Code.org materials. This research primarily relied on two theories of integration (i.e. Kiray, 2012. A new model for the integration of science and mathematics: The balance model. *Energy Education Science and Technology Part B: Social and Educational Studies*, *4* (3), 1181–1196) that provided insight into the level of interconnection between the disciplines and the relative amount of instructional time spent within each discipline. Findings revealed that cross-grade CS/CT concepts included sequencing, looping, and conditional logic. Within each category: (a) concepts were taught with increasing complexity across the grades, (b) the mathematics was dominant and CS/CT was important but secondary, and (c) three types of lessons emerged: No integration, partial integration, and full integration. Lastly, lessons generally included a transition from less integrated to more integrated activities with an initial focus on discipline-specific conceptual understanding prior to integrated activities.

ARTICLE HISTORY
Received 28 November 2017
Accepted 27 August 2019

KEYWORDS
Elementary school computer science; integrated curriculum; computer science lesson planning; computer science learning progressions; elementary mathematics

1. Introduction

Interest in K-12 computer science (CS) education has grown over the last several years as evidenced by national initiatives such as the 2016 White House call for Computer Science for All and the creation of the CS for All Consortium (http://csforall.org). This rapid growth has resulted in questions regarding how to implement CS and computational thinking (CT) across K-12. Should CS/CT be taught as stand-alone programs or integrated into existing subjects? If integrated, what are effective practices for teaching CS/CT in those contexts? While those supporting the CS for All movement agree about the benefit of this nascent initiative, questions still abound about what, how, and at what pace to teach. While Grover and Pea (2013) pointed out that the cognitive facets of children and computational neophytes learning programming was studied extensively in the 1980s, there is limited research on the efficacy of different approaches to CS/CT education.

While other disciplines have clearly established progressions for learning, such as those developed for mathematics education (e.g. Clements & Sarama, 2004; Confrey, Maloney, & Corley, 2014; Daro, Mosher, & Corcoran, 2011), such work is only emerging in the CS education literature (e.g. Rich, Strickland, Binkowski, Moran, & Franklin, 2017). Additionally, while a framework for K-12 CS education

CONTACT Maya Israel misrael@coe.ufl.edu

(https://k12cs.org/) and updated CS education standards (Computer Science Teachers Association, 2017) have been developed, there is little guidance for efforts to integrate these concepts into core areas such as mathematics. Thus, developing research-supported learning progressions for K-12 CS is a critical step for CS education.

To add complexity to understanding learning progressions in K-12 CS and CT instruction, there is a movement to teach CS and CT in an integrated manner within other disciplines (Jona et al., 2014; Lee, Martin, & Apone, 2014; Schanzer, Fisler, Krishnamurthi, & Felleisen, 2015). Reasons for focusing on integration include: (a) opportunities for introducing computing within real-world, authentic experiences that capitalize on CS/CT, and (b) greater access to CS/CT for underrepresented populations because integrated instruction occurs in general content areas rather than in opt in elective CS classes (e.g. Weintrop et al., 2016).

Papert (2006) anticipated "a widening focus from *how* people learn to *what* people learn" (p. 582). He used mathematics in particular as an example and hoped for curriculum that would "give children the means to find unique ways to create a personal mathematics of their own to love (p. 585)." A glimpse of this could be seen in the study by Harel and Papert (1990), who conducted a learning research project that had fourth grade students create instructional software to teach fractions. They found that students did better in both the mathematics and programming in the experimental group compared with the two controls. Thus, integrating computing into mathematics allowed Harel and Papert to study how computing could be used as the tool for constructing mathematics understanding.

1.1. Defining CS/CT in the context of this study

Although there are many definitions for computational thinking (CT), at its core, it can be defined as "a way of thinking that involves formulating problems, decomposing them, and structuring and communicating solutions so that humans can understand them and machines can process them" (Waterman et al., 2018, p. 283). The K-12 CS framework provides guidance on CT practices: (a) recognizing and defining computational problems, (b) developing and using abstractions, (c) creating computational artifacts, and (d) testing and refining computational artifacts. Brennan and Resnick (2012) applied CT practices that align with the Scratch programming language including: (a) experimenting and iterating, (b) testing and debugging, (c) reusing and remixing, and (d) abstracting and modularizing (Brennan & Resnick, 2012).

There has been a push to increase K-12 students' exposure to CT. This push is not new and goes back to Papert's (1980) assertion that computers can be used as tools for learning. Since Papert's description of CT in *Mindstorms*, the term CT has become quite loaded and has taken multiple meanings. Although elucidating the meaning(s) of CT is outside the scope of this paper, it is important to describe how this term was used within this study. For the purpose of this study, the terms CS and CT were used in tandem (i.e. CS/CT) to broadly describe experiences in which students engage in problem solving and algorithmic thinking alongside both programming and unplugged activities to engage with mathematics concepts.

1.2. Conceptual models for CS/CT content integration

In K-12 education, there is a move to integrate the subjects, given that they are often taught in isolation despite the fact that most subjects actualize in a cross-disciplinary manner (Masters, 2016). Several arguments have been made regarding the rationale for content integration including relevance to addressing real-world problems and highlighting the interconnections between disciplines (English, 2017). However, it is often noted that combining the disciplines in a manner that preserves the integrity of each of the disciplines often proves challenging and complex (English, 2017).

Several conceptional frameworks elucidate how content integration can occur. Vasquez (2015) stated that integrating science, technology, engineering, and mathematics (STEM) can create a

meta-discipline, defined as integration of separate subjects into a new area of study. Kiray (2012) described a process for understanding integration of mathematics and science through the relative focus of each discipline within instruction. Using Kiray's model, if CS/CT is integrated into mathematics wherein the mathematics had more emphasis than CS/CT, that instruction would be considered mathematics intensive, CS/CT connected; if the opposite were true, the instruction would be CS/CT intensive, mathematics connected. In another approach, Vasquez, Sneider, and Comer (2013) arranged integration along a continuum of activities: (a) no integration between content areas, (b) multidisciplinary instruction in which instruction is taught separately but references common themes, (c) interdisciplinary instruction where the disciplines are tightly linked, and finally, (d) transdisciplinary instruction in which skills across disciplines are applied to answer central driving questions. In this way, activities could be evaluated based on the level to which each individual disciplinary area is used within instruction. Finally, Bryan, Moore, Johnson, and Roehrig (2015) stated that content integration must be intentional as related to the content areas. They identified three types of integration: (a) integration wherein learning experiences have multiple interdisciplinary objectives, (b) integration wherein one content area supports another content area, and (c) integration wherein one content area serves as the context for learning another content area.

1.2.1. Mathematics and CS/CT integration

When it comes to content integration between mathematics and CS/CT, several authors point to the natural ways in which these disciplines can fit (Pei, Weintrop, & Wilensky, 2018; Sneider, Stephenson, Schafer, & Flick, 2014). Sneider et al. (2014) created a Venn diagram of mathematics and CT wherein common areas included problem solving, modeling, analyzing and interpreting data, and statistics and probability (Sneider et al., 2014). The authors explained that other areas are distinctly mathematical thinking (e.g. counting, geometry) or computational thinking (e.g. programming, data mining). Pei et al. (2018) similarly described the overlap between the two disciplines. They explained that CT and mathematics habits of mind are distinct areas that are mutually supportive and can be instructionally linked; however, opportunities to teach these areas together rarely happen. Weintrop et al. (2016) formulated a taxonomy wherein mathematics and CT could be integrated and included four categories: data practices, modeling and simulation practices, computational problem-solving practices, and systems thinking practices (Weintrop et al., 2016).

Despite the emerging work in integrating CT and mathematics, it can be challenging to find ways wherein the two disciplines can be taught both as distinctly authentic and in a manner that is meaningfully integrated. Additionally, as both Kiray (2012) and Vasquez et al. (2015) noted, one often finds differences between how teachers intend to integrate content and how content integration takes place in practice. There is a tension between the desire to create fully integrated, interdisciplinary or multidisciplinary experiences and maintain the focus on learning within each of the disciplines. Thus, these models can provide a useful lens for studying the integration of CS/CT into K-12 instruction.

1.3. Purpose of this study

Given the lack of research focused on how CS/CT can be integrated into elementary school settings, essential exploratory research involves describing the types of integrated activities created by elementary teachers who received professional development and had several years teaching CS/CT within their instructional settings. This study, therefore, explored the implementation of CS/CT in the context of elementary mathematics. This study had two major foci. First, this study investigated which CS/CT areas were emphasized in the lessons developed by the teachers. Second, this study examined the extent to which CS and CT were integrated within mathematics instruction through the lenses of both Kiray (2012) and Vasquez et al. (2013). This study was part of a larger research project funded by the National Science Foundation STEM + C program that seeks to develop learning trajectories for integrated elementary mathematics and CS/CT. The major aim of this study was to

shed light on initial attempts at integrating CS/CT across the grades in the context of elementary mathematics. Because of the early nature of this research, it is premature to design CT learning progressions based on this early work. Three research questions guided this study:

(1) Which CS/CT concepts were most notable in lesson plans developed by elementary teachers across grades 1–5?
(2) How did instruction of CS/CT change across the grades?
(3) In what ways was CS/CT integrated into mathematics instruction across grades 1–5?

2. Method

This study primarily made use of qualitative document analysis methodology (Bowen, 2009; Mayring, 2000) in which instructional lesson plans were analyzed to investigate the CS/CT content as well as the level to which CS/CT was integrated into elementary mathematics instruction as well as the CS/CT content most prevalent in those lessons across the grades. Member check procedures were implemented alongside this lesson plan analysis to clarify participants' aims in lesson development as well as to confirm or correct themes that emerged in the analysis.

2.1. Setting and participants

This study took place in a public Midwestern elementary school with a CS for All initiative. This school made financial and time investments toward bringing CS/CT into instruction to actualize its CS for All mission. These investments included purchase of equipment, hiring two CS/CT instructional coaches, working with a university for teacher professional development (PD) related to CS/CT, and providing time to develop integrated mathematics and CS/CT units. The school had a diverse student population that included 58% of students from low-income households, defined as households receiving public aid or eligible to receive free or reduced-price lunches. The student demographic makeup included 48% White/Caucasian, 25.5% African American, 7.8% Hispanic, 5.9% Asian and 12.8% other or multi-racial students. This school population was representative of other local schools and provided an opportunity to study CS/CT integration in a typical public elementary school with a wide range of learner backgrounds, abilities, and interests.

During the summer prior to the first year of the CS/CT initiative, approximately 20 teachers attended a week-long workshop at a local university in collaboration with the colleges of computer science and education. During the subsequent two years, teacher PD included embedded instructional CS/CT coaching and monthly after-school PD sessions. As this study was part of a National Science Foundation project, school personnel had access to university faculty with expertise in CS/CT and K-12 education as well as curriculum developers of their mathematics curriculum. The university team was available on an as-needed basis and during curriculum writing sessions to advise the teachers on integration, CS/CT, and possible connections between disciplines.

After obtaining Institutional Review Board approval, 13 teachers and two CS/CT coaches were recruited to participate in this study. These teachers ranged across 1st–5th grades. To develop the integrated units, the teachers and CS/CT coaches met for 10 sessions during the 2014–2015 school year to write initial lessons. They wrote integrated lessons in an online format through Google Docs so they could collaboratively write, edit, and share the lessons among group members. After an initial implementation in the 2015–2016 school year, the teachers and CS/CT coaches, with support from university and math curriculum professionals, refined the materials as they taught them and continued to write new materials through spring of 2017. Thus, the integrated CS/CT lessons were developed through an iterative development process.

During member-checking procedures, one of instructional coaches indicated that at the beginning of the project, prior to lesson implementation and refinement efforts, limited attention was

paid to CS/CT learning progressions across the grades because (a) none of the students had significant CS/CT exposure so they were all novices, (b) the teachers had limited understanding of what types of computing activities were appropriate across the grades, and (c) the teachers were still learning how to bring CS/CT into their mathematics instruction. Additionally, the CS coaches indicated that the initial considerations of integration involved bringing any CS/CT experiences into mathematics instruction. They indicated that as the lesson writers gained experience with integrated lesson development, they focused more on activities where students could demonstrate mathematical understanding through CS/CT activities. Lastly, as lesson development continued, the two CS/CT coaches developed and used a matrix of available scope and sequence documents from Code.org CS Fundamentals, Google CS First, and the CSTA standards. They used this matrix as a guideline for lesson development, focusing primarily on aligning computing concepts at each grade level to those that were emphasized in the Code.org CS Fundamentals curriculum that was used once a week for an hour as a stand-alone computing time. The instructional coaches further shared that as development continued, the writers felt it was important to align the computing concepts students engaged with during stand-alone and integrated CS/CT.

While this matrix included kindergarten, the writers of the integrated lessons chose not to write for that grade level. However, they indicated that they used the matrix, and what students were learning in Kindergarten during stand-alone Code.org lessons, to guide the first-grade integrated lesson writing. The teachers and CS/CT coaches also met regularly to discuss student progress, observations about students' computational skills, and challenges that students faced during different activities. Thus, by the second round of lesson plan writing, the teachers and CS/CT coaches more purposefully considered scope and sequence as well as opportunities for integration. Figure 1 showcases the process that teachers and CS/CT coaches undertook during the beginning phases of CS/CT integration planning. It was important to them to begin mapping preliminary learning progressions and goals at each grade level even though there was not a great deal of readily-available guidance for this work.

2.2. Curricular materials and software

The elementary mathematics curriculum used in this study was Everyday Mathematics, 4th edition (EM4; McGraw Hill). Everyday Mathematics is a "spiral" curriculum, or one in which there is an iterative revisiting of topics. Each time a topic is revisited, the instruction builds upon the previous visit (Harden, 1999). Thus, topics in a spiral curriculum are introduced and mastered over a series of instructional experiences rather than taught to mastery in individual lessons or units. The 4th edition of the research-based, Everyday Mathematics curriculum was updated to align with the Common Core State Standards, including the Standards for Mathematical Practice (SMP). These SMP's describe the types of expertise of "doing" that students should expect to develop over time as they engage in increasingly more complex mathematical activities. They describe the mathematical "processes and proficiency's" long essential to the act of engaging in mathematical thinking and doing (Standards for Mathematical Practice – Common Core State Standards Initiative, 2019). The lesson writers began by picking two instructional units per grade level within the EM4 curriculum for integration with CS/CT. One unit would be taught in the third quarter of the academic year and one in the fourth quarter. The third-grade writing team wrote an additional unit as well.

Lessons were created in the following areas:

- 1st grade: Addition word problems
- 2nd Grade: Geometry (properties of polygons)
- 3rd Grade: Fractions (fractional parts and fractions on the number line), multiplication, and number sense through number stories
- 4th Grade: Number Sense through number stories, Fractions (comparing)
- 5th Grade: Area and volume, Algebraic Thinking, Operations (multiplication and division)

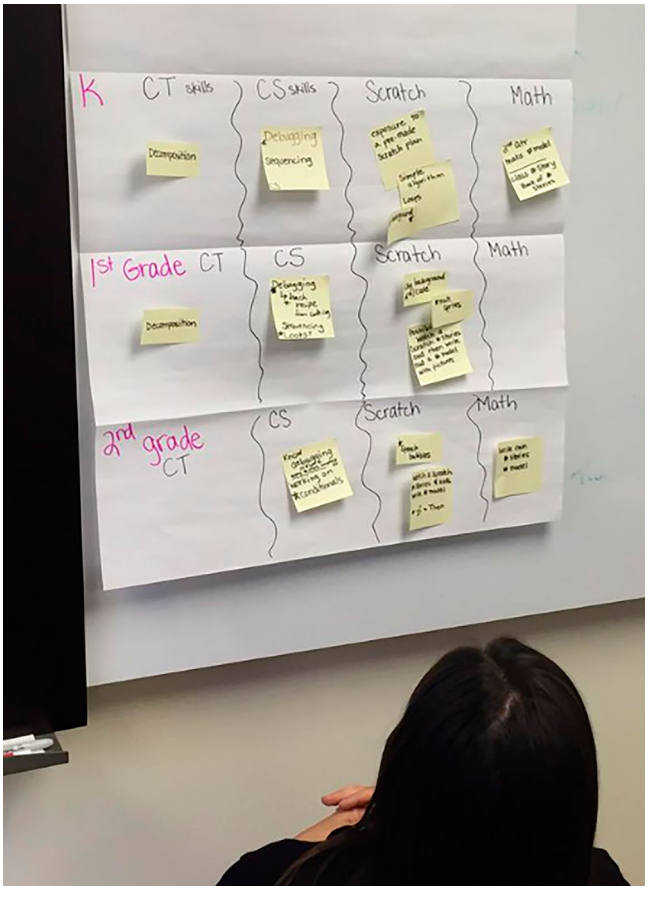

Figure 1. Planning integrated CS/CT lesson progressions.

The primary computing environment used for the integration of CS/CT and math in this study was Scratch (http://scratch.mit.edu), a block-based programming environment in which students drag and attach blocks of code to create animations and other interactive media (Maloney, Resnick, Rusk, Silverman, & Eastmond, 2010). Scratch was designed to allow students to progress to increasingly sophisticated programming within the same software (Grover & Pea, 2013; Resnick et al., 2009). Block-based programming languages such as Scratch are often said to be advantageous in that their use can reduce syntax errors in novice programmers (Begel, 1996). Furthermore, recent work by Weintrop and Holbert (2017) suggested that block-based languages such as Scratch that have visual command block choices alongside the program, allow users to see possibilities that they may not know exist. In other words, blocks allow the user to experiment in their programming, freeing them from both having to remember commands as well as helping to assuage fears related to syntax errors or issues of structure more commonly associated with text-based programs (Weintrop & Holbert, 2017).

A secondary computing environment was Code.org CS Fundamentals, which is also a block-based programming language but differs from Scratch in that it allows constrained choices of blocks with which to build programs based around sequential puzzle-based activities related to different computational topics. While not directly related to the integrated CS/CT work, students engaged in this stand-alone, self-guided work once a week in Code.org to build general programming knowledge. During integrated computing, however, Code.org CS Fundamentals materials functioned as precursor or supporting activities. Two types of Code.org activities were implemented in this way. The first were

unplugged activities that introduced or reinforced CS/CT concepts without the computer through hands-on activities. The second set of activities were coding puzzles that allowed students to engage with a concept more directly than within the integrated materials. Thus, Scratch was primarily used for integration and Code.org was primarily used for supporting or introductory activities. Typically, the teachers had a set amount of time for math instruction. This block of time varied between 45 min to one hour per day. The integrated mathematics and CS/CT instruction had to fit within this time frame. During member checking conversations, the teachers indicated that in developing the integrated materials, they were conscious about the time required to complete these integrated activities. With that said, the teachers also indicated that they were given flexibility in the schedule in case the lessons took a little longer than usual and some lessons were planned so that they spanned over the course of as many as three days. The EM4 curriculum, with which the CS/CT activities were integrated, also includes lessons that spanned more than one day. Furthermore, the structure of EM4, which also includes some multi-day lessons, enabled some flexibility in scheduling. The EM4 curriculum is built around four days a week of instruction with the fifth day being saved for reteaching or math games practice. Thus, the teachers treated this fifth day as flexible. All integration activities were generated by the teachers and the CS/CT coaches collaboratively. CS/CT areas that were highlighted included sequencing, looping (including nested loops), conditional logic, decomposition, debugging and to a lesser extent, variables and functions. Each of the integrated lessons addressed at least one of these computational concepts. Lessons can be found at https://ctrl.education.illinois.edu/ltec.

2.3. Data sources and analysis

The primary data analyzed were lesson plans developed and used by elementary teachers as they integrated CS/CT into their mathematics instruction. These lesson plans included model activities and computational artifacts in Scratch and Code.org CS Fundamentals, student planning documents, assessments and rubrics as well as a matrix of scope and sequence documents that was used in the second-round writing to assist in determining which computing concepts were to be focused upon. In total, 47 lesson plans across grades 1–5 were gathered for analysis.

2.3.1. Document analysis

Lesson plans were analyzed through a collaborative effort between researchers involved in curriculum writing as well as researchers independent from the curriculum writing in order to engage in frequent member checks related to lesson content and intent. Two qualitative methodologies were used for the document analysis. A deductive approach was taken to categorize lessons and activities based on computational concepts. A more inductive approach was initially taken to understand the level of integration. This inductive approach used a constant comparative process (Glaser & Strauss, 1967) until common themes related to integration were established.

The research team started with a small fragment of the data (i.e. first grade lesson plans). These data were inspected to develop a set of codes with the categories of computational concept and whether the lessons had any evidence of integration with mathematics. New data was then introduced (e.g. lessons from 2nd grade), and were compared to the first set of data. In this manner, the research team utilized deductive codes in the area of CS/CT concepts (i.e. sequencing, looping, conditionals, decomposition, and debugging) and a more inductive approach for degree of integration between mathematics and CS/CT. Through this data analysis process, the research team operationalized the three categories for level of integration: (a) No integration: Activities that taught disciplinary content separately (e.g. an unplugged activity that introduces the concept of conditional logic), (b) Partial integration: Activities in which content in one content area is used to reinforce content from another area or activities in which academic language from one content area is taught in the context of another content area (e.g. a teacher describing "debugging" a math problem), and (c) Full integration: Content from both content areas are taught and used in equal

Looping	Grade	Grade-specific explanation	Lesson Plan Codes
1A-A-5-2 Construc programs, to accomplish a task or as a means of creative expression, which include sequencing, events. and simple loops, using a block-based visual programming language, both independently and collaboratively **1A-A-3-7 Construct and execute an algorithm (set of step-by-step instructions) which includes sequencing and simple loops to accomplish a task, both independently and collaboratively, with or without a computing device.**	2	Looping explicitly introduced. Loops are introduced mostly through unplugged activities, but also through a stamping activity. These activities showcase that when a command is given a certain number of times, the program is executed that number of time. Efficiency is introduced as a means of creating a program with fewer steps or blocks. This is less work and takes less time to create.	-Whole group unplugged activity: Jumping jacks. Ex: Students do 4 jumping jacks and describe, repeated activity -(Use) Teacher modeling of stamping activity: When sprite clicked, switch backdrop, repeat stamp, move 30 steps. - Code.org "Getting loopy" unplugged activity: Shows efficiency in looping through dance - (Use) Unplugged activity: Use of laminated Scratch tiles of when sprite clicked, switch backdrop, repeat stamp, move 30 steps (Sequencing activity at the Use level)

Figure 2. Screen capture of looping code example.

measure (e.g. learning about polygons by creating a code to animate polygons). For each computational concept, a table was created with grade level, explanation of the computational concepts, and activities coded from the lesson plan. Figure 2 provides a screen shot of part of this coding document.

The research team met regularly to operationalize definitions and set up a consistent coding process. After collaboratively coding one lesson plan together, the research team coded another set of lesson plans independently.

In the end, the research team coded 47 different lessons. Within these lessons, the team identified 70 distinct activities that were coded for computing content and for level of integration. For example, in Figure 2, one activity from Code.org was called "Getting Loopy." This unplugged activity introduced efficiency as means of creating a program with fewer steps. It was coded for "Looping/Repetition" activity in the second grade. This activity was then coded as "No integration" as it taught the concept of looping/repetition outside of math instruction and did not refer to math content in the lesson plans. Later lessons leveraged this introductory unplugged activity in a more integrated manner (e.g. Walk a Polygon activity made use of sequencing and repetition). This analysis was then inputted into a Google spreadsheet tied to a web application called Awesome Table (https://ctrl.education.illinois.edu/ltec/database-of-research-lesson-materials). This spreadsheet and web interface included the following cells: (a) level of integration, (b) mathematics concept, (c) grade level, and (d) computing concept. Figure 3 provides a screen shot of the database interface.

Once lesson plans were individually analyzed, a secondary analysis examined computational concepts across grade levels. As stated above, for the purposes of this study, CS/CT are broadly described as experiences in which students engage in problem solving and algorithmic thinking alongside both programming and unplugged activities to engage with mathematics concepts. Therefore, the definitions of CT from the K-12 CS Framework were used alongside the Framework computing concept of Algorithms and Programming and its accompanying subconcepts (https://k12cs.org/). In this way, although there is disagreement in the field regarding definitions and implementation of CT in K-12, the research team attempted to use definitions consistent with conventions developed by the CS education researcher and practitioner communities. Once this analysis was completed,

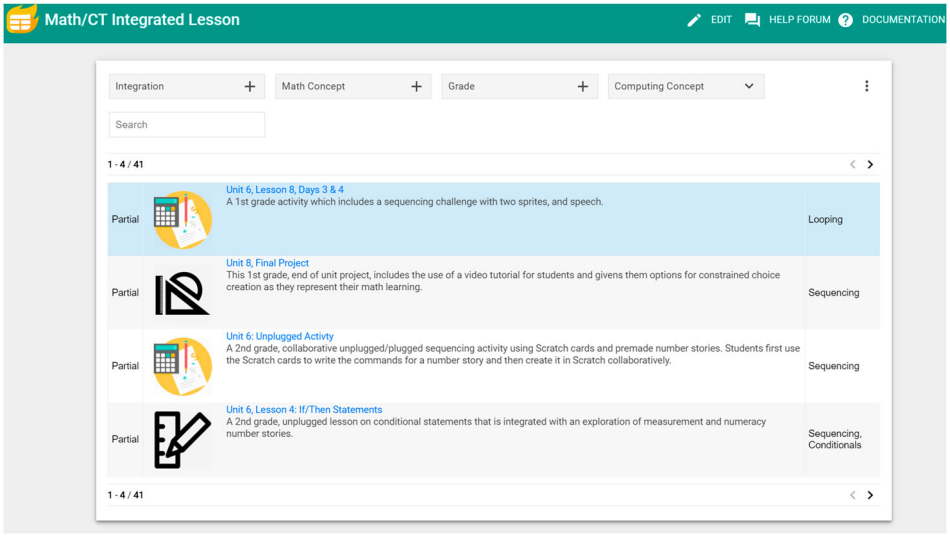

Figure 3. Screen capture of the awesome table database of lessons.

results were shared with CS university faculty as well as K-5 CS educators that were unrelated to this study for content and construct validation and to address questions about interpretation.

Once the research team had established codes for both computational concepts (through a deductive approach) and levels of integration (through an inductive approach), it was important to examine the consistency in which these codes were used. Instead of calculating percent agreement, which may result in inflated agreement scores due to researchers potentially picking the same codes by chance, Cohens Kappa coefficient (Cohen, 1960) was computed for interrater reliability across both the computing concepts and the three levels of integration. The Cohen's Kappa table was created across all codes in all lesson plans. Each of the 70 activities across the 47 lesson plans were coded for both computational concept and level of integration. Cohen's Kappa was computed across these codes and was 94% for computational concepts and 90% for level of integration.

Next, member check procedures took place to increase the credibility of the data analysis and involved asking participants whether research interpretations were correct in intent of both teaching the computational concepts and the integration that was intended. The research team also asked for clarification about intent of lesson design in order to better understand decisions about the progression of computational concepts and reasoning for the types of integration that was intended by the teachers. Lastly, qualitative themes related to computational concepts and levels of integration were then quantified for descriptive statistical analysis by ascertaining the numeric frequency of each code and subcode (Creswell & Plano Clark, 2011; Onwuegbuzie & Teddlie, 2003). These frequency counts were used to determine the prevalence of each theme (i.e. the number of lessons that addressed topics such as sequencing and looping and the level of integration with mathematics).

3. Results

Lesson plan analysis revealed that across grades 1-5, there were 47 lessons, split into two units per grade level, with the exception of third-grade, which included a third instructional unit. The number of lessons per unit varied greatly and ranged from a low of two lessons in one fifth grade unit, to a high of nine lessons in one second grade unit.

As mentioned above, across the grades, 70 distinct integration activities were noted across the lessons. For example, one lesson might have an unplugged activity followed by a plugged activity. In addition to these 47 lessons, teachers also covered general mathematics content from the EM4

teachers manual. These lessons were not included in the lesson plan analysis as they did not address integration. In fact, during member check interviews, teachers explained that there was a split between instruction that they considered integrated and instruction from the EM4 teacher's manual. Results were organized by research question below.

3.1. RQ1: Which CS/CT concepts were most notable in instruction across grades 1–5? and RQ2: How did instruction of CS/CT change across the grades?

The 47 lesson plans and accompanying 70 integration activities included a combination of CS/CT, mathematics, and integrated lessons. The most prevalent CS/CT concepts addressed were sequencing, looping, and conditional logic. To a lesser extent, decomposition, variables and functions were present, although there was less evidence of progressions within these areas. Interestingly, although the teachers discussed debugging a great deal during member check conversations, the lessons did not have explicit evidence of debugging activities. Rather, at the time of this study, debugging was introduced and modeled for the students as issues emerged. When asked about the lack of debugging explicitly evident in the lessons, the teachers indicated that in later lesson development, debugging would be more explicitly embedded in the lesson plans. Table 1 provides a general description of the emerging learning progressions within sequencing, looping, and conditional statements.

Five teacher-developed lessons focused solely on mathematics and did not contain any CS/CT content. These were, therefore, excluded from this portion of the analysis. The writers expressed that for the remaining 42 lessons, sequencing was seen as a fundamental aspect of creating programs or algorithms. While not overtly taught in all of these lessons, the authors felt the importance of order and precision was inherent. Building student understanding of the importance of when, how and which commands are executed was the sole CS/CT focus of many of the lessons at the earlier

Table 1. Computational concept across grades 1–5.

Grade	Description	Example Activity
1	Order matters when giving a computer commands. Some sort of an event is needed to begin or run code. First graders also play with the idea that one plans for expected outcomes for their program.	Making Speech Bubbles is an early lesson where students program conversations between 2 sprites. It emphasizes social skills ideas of one person speaking at a time and reinforces that order matters when speaking or writing just as when programming.
2	Students learn that precision and completeness are important in sequencing. Students learn to identify instances of repetition in code and ultimately learn that looping can be used to create the same shape with less steps. Simple if/then conditional logic is introduced.	Walking Out Polygons is an unplugged, collaborative activity that engages students in co-constructing an algorithm to walk out a polygon.
3	Use of the Use, Modify, Create structure, developed by Lee et al. (2011), allowed students to use and modify instances of conditional logic, looping and variables. Branching can allow users to interact with a program and make choices, variables have values that are dynamic and can add functionality, and branching can include *if, then and else* conditions.	Dance Remix provides students with an opportunity to first use and later modify a program wherein conditional logic is used to gather user input in order to make a sprite dance.
4	Creating programs requires considering both appropriateness of commands and order. Appropriateness can be decided only by considering the commands which came before and desired outcome of the program. Complexity is added as students use and modify programs, as well as create programs using *if/then/else* conditional logic.	*Ask a Question, Wait and Answer* involves pairs taking turns playing the role of questioner and responder. The idea of using conversation to explore sequencing (from 1st grade) continues. Students are guided towards asking questions, listening to the responder and using the response to drive further conversation via questioning.
5	Students explore multiple ideas in sequencing in fifth grade including the idea that different sets of instructions can produce the same result. The introduction of functions and variables increases the sophistication of the sequencing, while simplifying the code.	Willis Tower: Students utilize Unifix cubes, graph paper and Pseudo-Code to recreate the Willis Tower using functions. They examine how efficiency can be improved by implementing loops in their code.

grades. As lessons introduced increased sophistication of activities and programming tasks, sequencing was either combined with looping or conditional logic. Approximately 21% of the lessons (n = 10) had a combined focus on both sequencing and looping, while 27% (n = 13) included activities that included sequencing and conditional logic. Only two lessons, approximately 4%, combined sequencing, looping and conditional logic.

Data analysis revealed emerging learning progressions in the lessons across sequencing, looping, and conditional logic. These emerging learning progressions highlighted how, across the grades, the teachers and CS/CT coaches planned to provide increasingly more complex experiences to students through increasing the sophistication and functionality of their computational artifacts in two interconnected ways.

First, across the grades, the lessons and activities showed an increase in sophistication within each computing concept. For example, in the first-grade lesson plans, students were initially exposed to the idea that a programming sequence begins with an event and ends with an expected outcome. These first-grade lessons also aimed to teach that order matters when planning how to sequence commands, and that if that order is not followed, then the program may not work. During the second-grade lessons, students were taught to recognize repetition of code within the program's sequencing.

Later in second grade, the lesson plans added sophistication through activities that elucidated how using loops can simplify code. In another example, students in second grade were introduced to conditional logic through an unplugged if–then coding game. By the 4th grade, students were engaged in conditional logic activities in which they created fractional parts story problems using if/then/else blocks as well as operator and sensing blocks (see Figure 4).

Second, functionality and sophistication increased with the addition of, and intersection with, new computing concepts, such as when conditional logic is added to a program to allow for user input. For example, first grade lesson plans included limited focus on looping and no introduction to conditional logic. In second grade lessons, within the context of sequencing, students were also provided with instruction about how looping can increase efficiency and functionality of programs. Second grade lessons also introduced conditional logic through unplugged activities, but students were not expected to apply this knowledge to their own computational artifacts. By third grade, lessons provided opportunities for students to explore the idea that not all of their code is executed when they utilize conditional logic, or branching, in more complex programs. In fourth grade, lessons

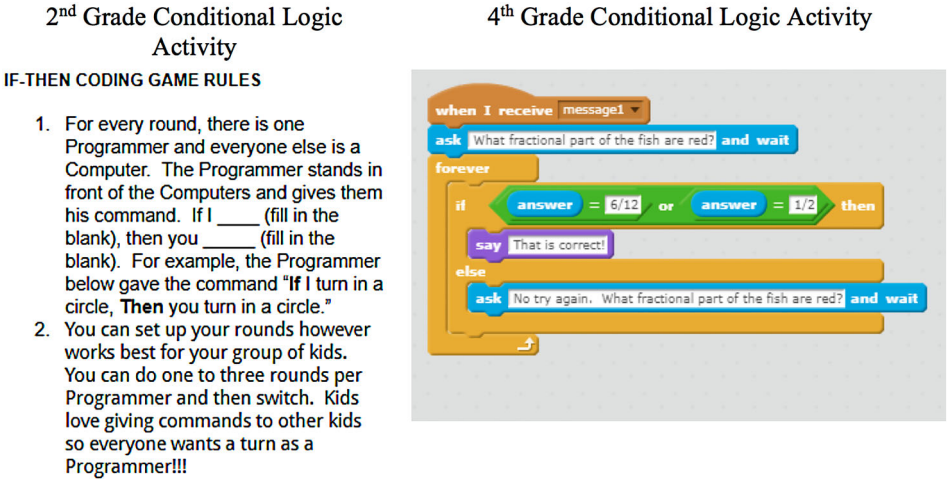

Figure 4. Progression of conditional logic activity.

included experiences wherein students applied knowledge of sequencing and conditional logic in creating their own programs.

Figure 5 illustrates the spike in the number instances of activities that address conditional logic within the third-grade lesson plans. Thus, while conditional logic was first introduced briefly in the second-grade lesson plans, this computational concept was more heavily emphasized in the third and fourth grades in conjunction with more sophisticated programming activities.

Moreover, while looping and conditional logic were noted in second through fourth grade lesson plans to add increased complexity to students' computational artifacts, instantiation of these ideas was relatively low in fifth grade lessons. Both procedures (i.e. functions) and variables, however, were introduced in fifth grade to allow for increased functionality in students' programs. Thus, as emphasis shifted from looping and conditional logic to functions and variables, there was a reduction in lessons emphasizing initial concepts that were emphasized in the earlier grades.

3.2. RQ3: In what ways was CS/CT integrated into mathematics instruction across grades 1–5

To understand how the lessons integrated mathematics and CS/CT, lessons were categorized by the relative amount of disciplinary content taught using Kiray's (2012) framework as well as level of integration using Vasquez's et al. (2013) framework. Analysis revealed that mathematics was taught to a greater extent than CS/CT content. In fact, approximately 86 lessons (65%) were taught directly from the EM4 teachers manual. During member-checking, when asked about the reliance on the teachers manuals, teachers indicated reasons included: (1) commitment to teaching the curriculum with fidelity, (2) need to cover the mathematics content to prepare students for district and state assessments, and (3) belief that the mathematics curriculum was an effective way to teach mathematics concepts.

When looking at the teacher-developed lesson plans, three levels of integration emerged: (a) No integration (i.e. teacher-developed mathematics or CS/CT lessons taught in isolation), partial integration (i.e. lessons in which CS/CT was used to reinforce mathematics content or mathematics was used to reinforce CS/CT content), and (c) full integration (i.e. mathematics and CS/CT content were taught together or the affordances of the CS/CT or mathematics were used to teach the

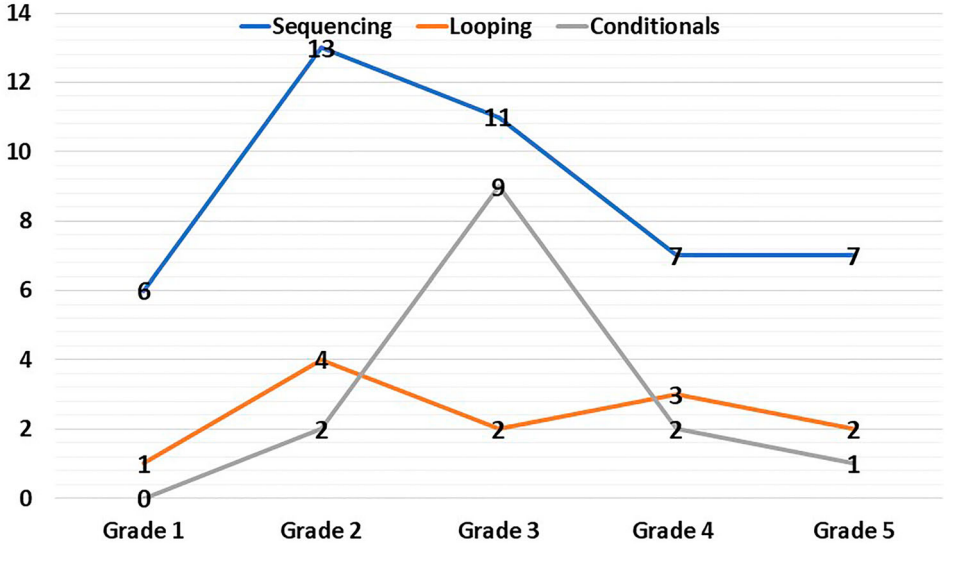

Figure 5. Sequencing, looping, and conditional logic at each grade level.

other subject). Figure 6 provides a visual representation of these three types of activities. The lesson plans revealed that there was often a cycle of mathematics lessons followed by CS/CT lessons followed by more integrated lessons. In this way, new disciplinary content was given enough emphasis prior to expecting students to use that knowledge within integrated contexts. Member-check interviews with both of CS/CT coaches and teachers revealed that they were cautious of teaching new content across both mathematics and CS/CT because of issues of increased cognitive load demands when teaching the CS/CT at the same time as the mathematics curriculum.

3.2.1. No integration

The majority of the lessons did not include any integration between mathematics and CS/CT. In addition to the 86 lessons from the EM4 teacher's manual, the teachers and CS/CT coaches wrote 22 lessons (46.8%) that either focused on mathematics ($n = 5$) or CS/CT ($n = 17$). The lesson plans that came directly from the EM4 teacher's manual were referred to in the Google Doc lesson plans through language such as, "Math Boxes 8.1, See pg. 695 in the teachers manual" (Second grade integrated unit). These isolated or supporting lessons that the teachers and CS/CT coaches wrote typically provided a pre-requisite experience to an integrated mathematics and CS/CT activity. In the second-grade unit, for example, in addition to the EM4 lessons, in a unit about attributes of different polygons, the lesson plan included an activity that began with the teacher reading aloud the book, *The Greedy Triangle* by Marilyn Burns (1994). This book was followed by an activity in which students look around the classroom to find and describe shapes similar to those from the book.

On the other hand, CS/CT lessons came from a variety of places including teacher-created mini lessons highlighting a specific computational concept or Scratch block and unplugged activities or from Code.org. According to the goal statements within these lesson plans, the supporting CS/CT lessons typically served the purpose of pre-teaching CS/CT skills in preparation for more integrated lessons. An example of a CS/CT lesson was a 4th grade unplugged activity that introduced the Scratch blocks "Ask a question and wait" and "answer" in the Sensing tab. In this activity, the students engaged in asking each other questions, waiting, and then responding to the question.

3.2.2. Partial integration

Fourteen of the analyzed lessons (29.7%) had some degree of integration between mathematics and CS/CT. In these lessons, mathematics content was more heavily emphasized and CS/CT reinforced that mathematics content. The primary focus of these lessons was the mathematics, but CS/CT activities were built into these lessons. In these lessons, CS/CT offered opportunities for students to demonstrate understanding of mathematics concepts by creating representations of mathematics. For example, in one 4th grade lesson, the lesson plan described providing opportunities for students to animate number stories as part of their mathematics instruction.

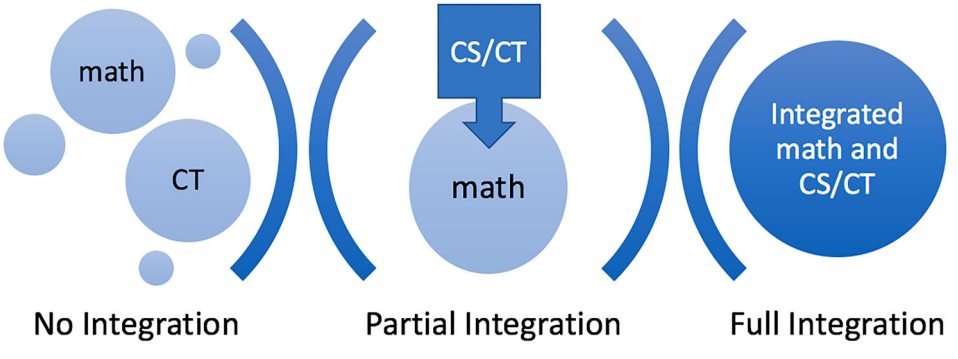

Figure 6. Three types of lessons in the integrated mathematics and CS/CT units.

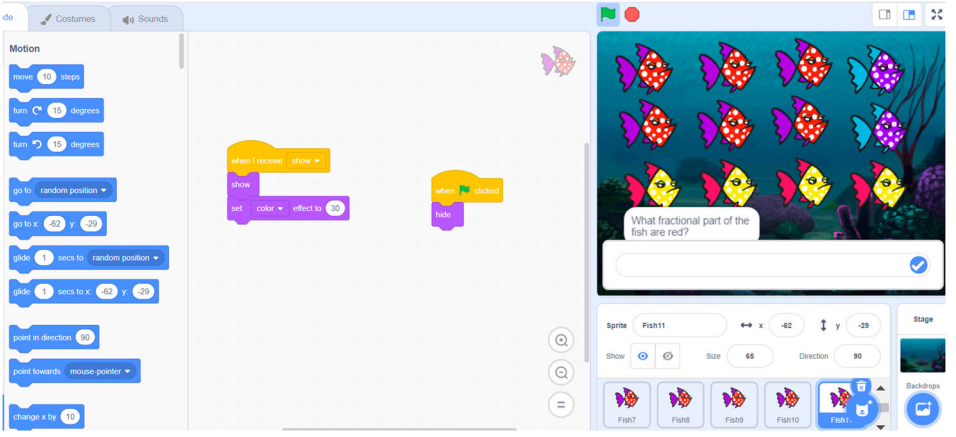

Figure 7. Teacher's display version of the fractional parts story problem.

This lesson focused on reinforcing understanding of fractional parts through the CS/CT activity (see Figure 7). The lesson plan began with a teacher model that showed the students how to animate a fractional part of a collection of items. The lesson then progressed to an activity wherein students create their own fractional part stories with animations in Scratch. The teacher model from the lesson plans was:

(1) There are 12 fish.
(2) Four of the fish are green, three are red, and the rest are blue.
(3) What fractional part of the fish are blue?

Thus, the majority of partial integration activities provided students with opportunities to demonstrate their mathematical understanding in Scratch. These activities typically provided for opportunities in which students could create the scenarios in which their demonstrated their understanding, but the mathematical content was well-defined.

3.2.3. Full integration

Eleven lessons (23.4%) were categorized as fully integrated. These lessons primarily taught mathematics concepts directly through CS/CT activities rather than teaching mathematics lessons and then having students demonstrate understanding through CS/CT. For example, a 2nd grade lesson plan indicated that students would learn about different polygons by animating those polygons in Scratch. The lesson plan explained that the students would walk the shapes of the polygons and then code those shapes. Figure 8 showcases the unplugged portion of the "walk a polygon" activity.

In another example, a 5th grade unit included full integration of mathematics and CS/CT as students learned to calculate area and volume by calculating the volume of the Willis Tower in Chicago. The EM4 teacher's manual included this activity, but the teachers added experiences in which students worked in pairs to create physical models of the Willis tower in Unifix cubes.

One student developed the model, which was hidden from the second student. They then wrote pseudo-code for the second student, who followed the code and worked as if they were a robot to build the tower represented by the code. If the structure was not completed correctly, the students then decomposed the code and went through a debugging process to fix it and try again. Next, the students switched roles. In the second phase of the lesson, the students learned about how functions could be used to simplify their pseudo-code and therefore the number of movements required by the "robot". The CS/CT concept of functions was a key component of the activity as students created

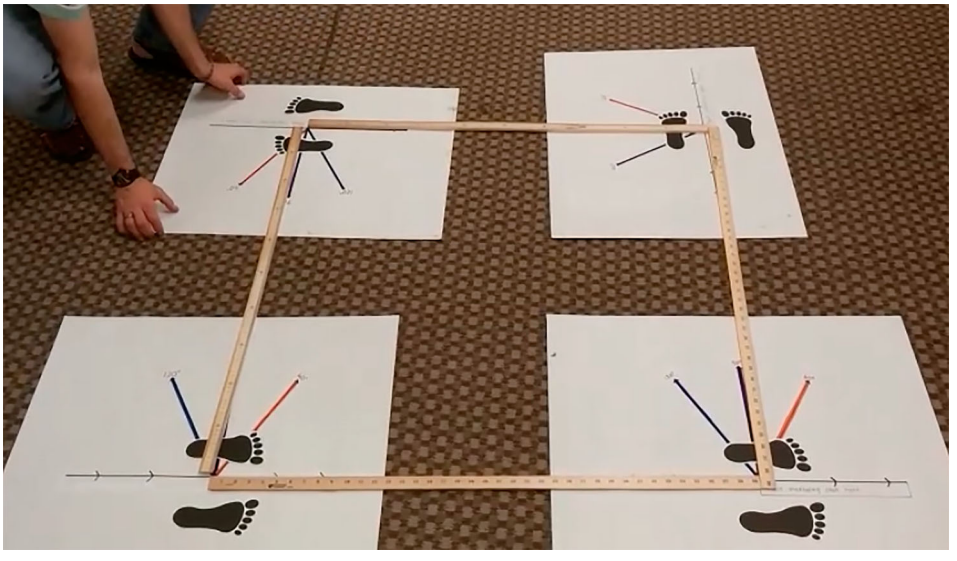

Figure 8. Unplugged "Walk a Polygon" activity as initial exploration the CS/CT concept that computers follow exact instructions.

pseudo-code to indicate the volume of the Willis Tower pieces. In these lessons, consequently, there was proportional attention given to both the mathematics and CS/CT content.

Thus, across all the units, there was a mix of activities that highlighted a range of computational concepts through a range of integration activities.

4. Discussion

This study showcased how elementary teachers and CS/CT instructional coaches, with professional development and support from CS and education university faculty, attempted to develop integrated mathematics and CS/CT instructional units across grades 1–5. Because of the emphasis on bringing CS/CT into the early grades, research into different implementation models is crucial. Given the limited research available to inform teachers and curriculum developers in how to integrate CS/CT into elementary mathematics (Rich et al., 2017), this implementation study provides a critical perspective. Findings from this study address both emerging progressions of computational concepts across the grades as well as approaches to integration.

4.1. Emerging CT learning progressions in the integrated lessons

Integrated lesson plans included an emerging progression of learning activities from first to fifth grades related to the areas of sequencing, looping, and conditional logic in what Duschl, Schweingruber, and Shouse (2007) described as successively more sophisticated ways of thinking about a topic (p. 219). These progressions are considered emerging as they are not fully developed; rather, they show a means of teaching CS/CT in increasing sophistication across the grades using the existing materials, resources, and supports within one instructional context. Rather, these emerging progressions should be expanded to year-long integrated experiences and then tested in multiple settings across students with different experiences in order to continue to refine the development of the CS/CT skills across the grades.

An interesting finding from the CS/CT emerging progressions was that the CS/CT concepts were not linear. Rather, as lessons increased in complexity, they called on other computational concepts. For example, as lessons included activities with increasing sophistication of sequencing concepts, those lessons began to integrate looping and conditionals. This finding about the

interconnectedness of computational concepts supports findings by Rich et al. (2017), who examined implicit learning progressions in CS educational studies and also found interconnectedness of CS and CT concepts across these studies. Although these authors stated that most of the literature they found addressed individual CS or CT learning goals, when examining the literature as a whole, they found a pattern of interconnectedness between CT concepts related to sequencing, looping, and conditional statements. Two additional computing concepts were seen in the fifth-grade lesson plans. These concepts, namely functions and variables, did not lend themselves to describe progressions as they only occurred in a limited number of lessons and were only at this one grade level. These lessons did, however, serve to increase the sophistication of the sequencing required in each lesson. Further, it is notable that even though the 2017 CSTA Standards were not complete at the time of the lesson development process in which the teachers engaged, the inclusion of teaching variables and functions (without parameters) at the fifth-grade level is consistent with the guidelines laid out in the those standards.

Another CS/CT area that did not lend itself to developing progression was debugging. Interestingly, although teachers across the grades discussed debugging and strategies that they used to help students debug their projects, debugging was not explicitly seen in the lesson plans. We anticipated finding specific strategies embedded in the lesson plans and were hoping to explain how the lessons addressed debugging across the grades. The teachers did indicate, during member check interviews, that future lesson plans will have an explicit focus on debugging so that the instructional practices that were utilized by the teachers would be noted in the lesson plans. Future research should therefore investigate how to teach debugging across the grades.

This study also highlighted the use of foundational, supporting CS/CT activities that were not designed as integrated activities, but rather were designed to provide prerequisite skills prior to integrated activities. At times, new computational concepts were introduced with concrete unplugged experiences in the form of physically acting out ideas (e.g. Students walking the shapes of polygons, see Figure 8) or through simple computing activities through Code.org CS Fundamentals or Scratch. Member-check interviews revealed that these isolated computing activities seemed to serve dual purposes of increasing sophistication of programming activities while decreasing cognitive load. Later lessons built on these formative lessons, giving students a chance to learn new ideas prior to computer-based activities.

Member-check interviews with CS/CT coaches also revealed that the writers utilized a matrix of available scope and sequence documents as well as a previous generation of the CSTA standards. Given that the K-12 CS Framework (2016) and the updated CSTA standards were not developed at the time of initial lesson creation, future research should investigate how learning progressions in the K-12 CS Framework can inform the development of future CS/CT curricula, including those that integrate into other disciplines such as mathematics.

4.2. Integration of CS/CT into elementary mathematics

The teachers and CS/CT coaches who developed the integrated units relied heavily on the EM4 mathematics curriculum but also created opportunities for students to both demonstrate their mathematical understanding through CS/CT activities and learn mathematics through CS/CT experiences. These lessons, therefore, parallel the commitment of other researchers who acknowledge the affordances of teaching CS/CT in the context of other disciplines (e.g. Jona et al., 2014; Lee et al., 2014).

In evaluating the integration of CS/CT into elementary mathematics, lesson plan analysis revealed that integration involved a range of activities from fully integrated to isolated or supporting mathematics and CS/CT lessons. In fact, although there were lessons that fully integrated the content areas, the majority of lessons did not include any integration of the two content areas. As stated above, the teachers explained that it was important to adhere to the mathematics curriculum to teach the content with fidelity. This finding supports English (2017) who explained how challenging it can be to integrate content areas while at the same time preserving the integrity of the disciplines.

Additionally, all integrated lessons had to be created by the teachers. Even though they had support from content experts and CS/CT coaches, there was no existing integrated mathematics and CS/CT curriculum for them to implement. Rather, they had to plan, create, and iterate themselves. Future research should investigate whether teachers would use more integrated lessons if those lessons were available to them as compared to the present study, wherein they had to create all the integrated materials themselves.

The theoretical frameworks offered by Kiray (2012) as well as Vasquez et al. (2013) provided a useful frame for interpreting the lesson plans. Kiray's model addressed the amount of content taught across disciplinary areas. Given that the analyzed integrated lessons primarily focused on the mathematics, Kiray's (2012) balance model would classify them as "math centered, CS/CT assisted". Kiray explained the reason for the dominance of isolated, disciplinary instruction was that this type of instruction was the primary mode of instruction. That is, most content is not taught in an interdisciplinary manner; even when attempting to integrate content areas, it is difficult to do so because of the nature of disciplinary instruction.

- Disciplinary: Lessons taught in isolation in a disciplinary manner where students learn concepts of separate disciplines separately from each other. Most lessons fell into this category, either as a means of conveying the math concepts or providing context in CS/CT to be used in future instruction.
- Multidisciplinary: Content areas taught separately but in reference to common themes. These lessons typically referred to other content areas (e.g. math lessons that reference CS/CT or CS/CT lessons that reference mathematics). For example, there was a math lesson on decomposing fractions that referred to decomposition in CS, but did not provide additional instruction beyond pointing out the connection between the disciplines.
- Interdisciplinary: Concepts and skills from two or more disciplines taught in a linked manner. Lessons that included integration typically fell within this category. These lessons fell along the continuum described in Kiray (2012) wherein one content area may have been more heavily emphasized but the two content areas were linked and taught together.
- Transdisciplinary: Knowledge from two or more disciplines used to provide students with authentic problem- and project-based learning opportunities. There was no evidence of this type of integration in the lessons.

In using Vasquez and colleagues' model, most lessons fell in the disciplinary category (no integration). However, 14 lessons included partial integration and 11 lessons fully integrated content. These integrated 25 lessons had content that would be considered either multidisciplinary or interdisciplinary in the STEM integration model. Within these lessons, the teachers organized instruction in an interdependent and interconnected manner where students would apply understanding from both mathematics and CS/CT. Vasquez (2015) explained barriers to transdisciplinary instruction as this type of instruction takes extensive resources, collaboration, and time. Thus, the lack of transdisciplinary activities in the analyzed lessons was consistent with Vasquez' assertions.

Another implication of the reliance on isolated or supporting CS/CT activities was that at the time of this study, the elementary students had limited previous exposure to CS/CT. Thus, the supporting CS/CT lessons may have been a necessary step that allowed students to build computational skills in a scaffolded manner in preparation for lessons that integrated mathematics and CS/CT. It may be that in subsequent years, and with additional computing experience, only lesson plans at the earliest grades will need the high percentage of precursor lessons found in the analyzed lesson plans, making more room for truly integrated and coherent computing within the curriculum. On the other hand, it could also be that as CS/CT concepts increase in sophistication, there will continue to be a need for supporting precursor activities. Moreover, while some students may not need isolated, supporting lessons, due to learner variability, some students may gain extensive benefit from their use. Precursor or supporting lessons may also prove to be a tool teachers wield in assisting new students or

those lacking comparable computing experience to the remainder of the class. Future research should examine the evolution of integrated lesson materials as students become more proficient in the CS/CT at earlier ages. It is an empirical question whether there will be a decreasing need for such activities in the future as CS for All takes hold in schools or whether as students gain additional expertise, the precursor supporting activities will involve more complex CS/CT content.

The curricular context in which lessons were integrated is also of note. Both the supporting and integrated lessons were written to work within a spiral mathematics curriculum. A spiral curriculum is one in which there is an iterative revisiting of topics, wherein each visit builds upon the previous one Harden (1999). Thus, topics in a spiral curriculum are introduced and mastered over a series of instructional experiences rather than taught to mastery in individual lessons. Several questions emerge about integrating CS/CT into a content area that is taught through a spiral curriculum method. First, CS/CT learning progressions are only now being developed, so there is no guidance in how to spiral curricular materials. Teachers, therefore, do not have a clear way of knowing how long to remain on key CS/CT concepts, how these key concepts should be revisited, and when to move onto new concepts. Second, there is little guidance in how to consider integrating CS/CT that is not taught in a spiral manner into a content area that is taught through a spiral approach. Thus, several implications for future research in this context exist including (a) a comparison of the barriers and affordances of integrating CS/CT into spiral versus non-spiraled curriculum, and (b) an exploration of ways in which spiraling the learning of CS/CT concepts can occur in the context of integrating CS/CT into core curriculum.

Lastly, the lesson writers expressed two major considerations related to curriculum and integration. In trying to find opportunities for integrated lessons, the writers stated that it was often hard to find a good match between specific mathematical content and CS/CT ideas. When facing this challenge, they looked to the Standards for Mathematical Practice (SMP) for further opportunity as they saw certain synergies between the SMPs and the computational practices which they were trying to build into these integrated experiences. The emphasis of the Standards for Mathematical Practice differ from the Standards for Mathematical Content. The ideas behind the former rest on a long history of delineating the thinking processes which proficient mathematical thinkers engage as they go about solving mathematical problems (Cuoco, Goldenberg, & Mark, 1996; Ferrini-Mundy & Martin, 2000; National Research Council, 2005). For these mathematical practices to become habit, or tools that students will use on their own, it is necessary to provide multiple opportunities to explore and utilize the SMPs. The SMPs include the following: (a) make sense of problems and persevere in solving them, (b) reason abstractly and quantitatively, (c) construct viable arguments and critique the reasoning of others, (d) model with mathematics, (e) use appropriate tools strategically, (f) attend to precision, (g) look for and make use of structure, and (h) look for and express regularity in repeated reasoning. Member check discussions with teachers revealed that they acknowledged that as with the computational practices, students should not be expected to master the SMPs in short order, but instead engage in their use over the course of many opportunities, spanning multiple years. Therefore, future research should investigate planning related to the teaching of the SMPs (and computational practices) at multiple levels (e.g. within units, across grade levels, across the school year) to ensure the sufficient opportunity to engage and master these proficiencies (Mateas, 2016).

4.3. Limitations

Findings from this study need to be considered in light of several limitations. First, as with all document analysis studies, the documents analyzed within this study cannot provide complete context (Bowen, 2009). Although the lesson plans analyzed provided information about the content the teachers and CS/CT coaches considered important to teach within integrated CS/CT and mathematics lessons and member-check procedures allowed for verification of interpretation by the participants, the lesson plans only provide written evidence of lesson development rather than lesson implementation. Second, the units developed did not represent a full year-long curriculum. Because of this constraint

on lesson development, the integrated units could only address mathematics content taught at the time of lesson implementation. Therefore, the teachers and CS/CT coaches had to find ways to fit CS/CT into existing mathematics instruction rather than consider integration from the inception of curriculum development. Although this type of instructional barrier would be common across most elementary teaching scenarios, findings related to integration and CS/CT taught could perhaps be different if the teachers had more flexibility in terms of the mathematics taught. Finally, as noted in the methods, the school in which this study took place had made a concerted effort to integrate CS/CT into the curriculum over several years. Therefore, the resources, supports (e.g. coaches), planning time available to create an idea scenario for elementary mathematics and CS/CT integration, and dedicated instructional time may not be representative of schools without a CS/CT instructional focus. Because of this significant level of support, there are implications for transferability to less resourced schools.

4.4. Conclusion

This study provided a lens into what it would take to design integrated instructional units for elementary classrooms in the context of mathematics and CS/CT instruction. Results highlighted the multiple layers of complexity of designing such instruction and result in multiple questions that should be investigated in future studies. Example questions include: (a) Was the level of integration a product of teachers need to follow the mathematics curriculum with fidelity? and (b) Did isolated, supporting lessons build sufficient foundational knowledge in both mathematics and CS/CT to allow for a more authentic integration? Given the proliferation of CS/CT activities in elementary schools, it is critical to continue investigating multiple models, instructional delivery approaches, challenges, and affordances of bringing CS/CT into elementary education classrooms.

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

This work was supported by National Science Foundation [STEM+C Award Numbers: 1742466 and 1932920].

Notes on contributors

Maya Israel, Ph.D. is an associate professor of educational technology in the School of Teaching and Learning at the University of Florida. Her research and outreach efforts focus on increasing access and engagement for academically diverse learners in K-8 computational thinking and computer science education. As the research director of the Creative Technology Research Lab, Dr. Israel conducts research on several National Science Foundation projects focused on integrating computing into elementary education as well as accessible computer science education. Lastly, she also consults with school districts on the development and implementation of computer science instruction that meets the needs of all learners, including those with disabilities.

Todd Lash is a doctoral student in Special Education at Creative Technology Research Lab at the University of Illinois. Todd's research interests include increasing the equity in and access to high-quality computer science education for all students. He studies the integration of computer science into K-5 curricula and how Universal Design for Learning (UDL) may be used as a way to engage all learners. Todd serves on the advisory boards of multiple nationally scaled computer science education projects and has conducted extensive outreach, consultation, and professional development with the CS education community.

ORCID

Maya Israel http://orcid.org/0000-0003-0302-6559

References

Begel, A. (1996). *Logoblocks: A graphical programming language for interacting with the world* (pp. 62–64). Boston, MA: Electrical Engineering and Computer Science Department, MIT.

Bowen, G. A. (2009). Document analysis as a qualitative research method. *Qualitative Research Journal*, *9*(2), 27–40.

Brennan, K., & Resnick, M. (2012, April). *New frameworks for studying and assessing the development of computational thinking*. Proceedings of the 2012 annual meeting of the American Educational Research Association, Vancouver, BC (Vol. 1, p. 25).

Bryan, L. A., Moore, T. J., Johnson, C. C., & Roehrig, G. H. (2015). Integrated STEM education. In C. C. Johnson, E. E. Peters-Burton, & T. J. Moore (Eds.), *STEM road map* (pp. 23–37). New York, NY: Routledge.

Burns, M. (1994). *The greedy triangle*. New York, NY: Scholastic.

Clements, D. H., & Sarama, J. (2004). Mathematical thinking and learning trajectories in mathematics education learning trajectories in mathematics education. *Mathematical Thinking and Learning*, *6*(2), 81–89.

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

Computer Science Teachers Association. (2017). *CSTA K-12 computer science standards, revised 2017*. Retrieved from https://www.csteachers.org/page/standards

Confrey, J., Maloney, A. P., & Corley, A. K. (2014, October). Learning trajectories: A framework for connecting standards with curriculum. *ZDM Mathematics Education*, *46*(5), 719–733. doi:10.1007/s11858-014-0598-7

Creswell, J. W., & Plano Clark, V. L. (2011). *Designing and conducting mixed methods research*. Los Angeles, CA: SAGE Publications.

Cuoco, A., Goldenberg, E. P., & Mark, J. (1996). Habits of mind: An organizing principle for mathematics curricula. *The Journal of Mathematical Behavior*, *15*(4), 375–402.

Daro, P., Mosher, F. A., & Corcoran, T. B. (2011). *Learning trajectories in mathematics: A foundation for standards, curriculum, assessment, and instruction* (CPRE Research Reports). Retrieved from https://repository.upenn.edu/cgi/viewcontent.cgi?article=1019&context=cpre_researchreports

Duschl, R. A., Schweingruber, H. A., & Shouse, A. W. (Eds.). (2007). *Taking science to school: Learning and teaching science in grades K-8* (Vol. 49, No. 2, pp. 163–166). Washington, DC: National Academies Press.

English, L. D. (2017). Advancing elementary and middle school stem education. *International Journal of Science and Mathematics Education*, *15*(1), 5–24.

Ferrini-Mundy, J., & Martin, W. G. (2000). *Principles and standards for school mathematics*. Reston, VA: National Council of Teachers of Mathematics (NCTM).

Glaser, B. G., & Strauss, A. L. (1967). Grounded theory: The discovery of grounded theory. *Sociology, the Journal of the British Sociological Society*, *12*, 27–49.

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

Harden, R. M. (1999). What is a spiral curriculum? *Medical Teacher*, *21*(2), 141–143.

Harel, I., & Papert, S. (1990). Software design as a learning environment. *Interactive Learning Environments*, *1*(1), 1–32.

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

Kiray, S. A. (2012). A new model for the integration of science and mathematics: The balance model. *Energy Education Science and Technology Part B: Social and Educational Studies*, *4*(3), 1181–1196.

Lee, I., Martin, F., & Apone, K. (2014). Integrating computational thinking across the K-8 curriculum. *ACM Inroads*, *5*(4), 64–71.

Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., … Werner, L. (2011). Computational thinking for youth in practice. *ACM Inroads*, *2*(1), 32–37.

Maloney, J., Resnick, M., Rusk, N., Silverman, B., & Eastmond, E. (2010). The scratch programming language and environment. *ACM Transactions on Computing Education*, *10*(4), 1–15. doi:10.1145/1868358.1868363

Masters, G. (2016). *Policy insights: Five challenges in Australian school education*. Melbourne: Australian Council for Educational Research. Retrieved from https://research.acer.edu.au/cgi/viewcontent.cgi?article=1004&context=policyinsights

Mateas, V. (2016). Debunking myths about the standards for mathematical practice. *Mathematics Teaching in the Middle School*, *22*(2), 92–99.

Mayring, P. (2000). Qualitative content analysis. *Forum: Qualitative Social Research*, *1* (2), Art. 20. Retrieved from http://www.qualitative-research.net/index.php/fqs/article/view/1089/2385#gcit.

National Research Council. (2005). *On evaluating curricular effectiveness: Judging the quality of K–12 mathematics evaluations*. Washington, DC: National Academy Press.

Onwuegbuzie, A. J., & Teddlie, C. (2003). A framework for analyzing data in mixed methods research. In A. Tashakkori & C. Teddlie (Eds.), *Handbook of mixed methods in social & behavioral research* (pp. 397–430). Thousand Oaks, CA: Sage Publications.

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

Papert, S. (2006). Afterword: After How comes what. In R. K. Sawyer (Ed.), *The Cambridge handbook of the learning sciences* (pp. 581–586). Cambridge: Cambridge University Press.

Pei, C., Weintrop, D., & Wilensky, U. (2018). Cultivating computational thinking practices and mathematical habits of mind in lattice land. *Mathematical Thinking and Learning*, *20*(1), 75–89.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., … Kafai, Y. B. (2009). Scratch: programming for all. *Communications of the ACM*, *52*(11), 60–67.

Rich, K. M., Strickland, C., Binkowski, T. A., Moran, C., & Franklin, D. (2017, August 18–20). *K-8 learning trajectories derived from research literature: Sequence, repetition, conditionals*. Proceedings of the 2017 ACM conference on International Computing Education Research, Tacoma, WA (pp. 182–190). New York, NY: ACM.

Schanzer, E., Fisler, K., Krishnamurthi, S., & Felleisen, M. (2015, March 4–7). *Transferring skills at solving word problems from computing to Algebra through bootstrap*. Proceedings of the 46th ACM technical symposium on Computer Science Education, Kansas City, MO (pp. 616–621). New York, NY: ACM.

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

Standards for Mathematical Practice. Common Core State Standards Initiative. (2019). Retrieved October 11, 2019, from Corestandards.org website: http://www.corestandards.org/Math/Practice/

Vasquez, J. A. (2015). STEM—Beyond the Acronym. *Educational Leadership*, *72*(4), 10–15.

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

Waterman, K., Goldsmith, L., Pasquale, M., Goldenberg, E. P., Malyn-Smith, J., DeMallie, A., & Lee, I. A. (2018). Integrating computational thinking into elementary mathematics and science curriculum materials and instruction. Pixel (Ed.), *Conference proceedings: The Future of Education 2018*. Florence: Libreria Universitaria Edizioni.

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.

Weintrop, D., & Holbert, N. (2017, March 8–11). *From blocks to text and back: Programming patterns in a dual-modality environment*. Proceedings of the 2017 ACM technical symposium on Computer Science Education, Seattle, WA (pp. 633–638). New York, NY: ACM.