Article

Computational Thinking Equity in Elementary Classrooms: What Third-Grade Students Know and Can Do

Journal of Educational Computing Research
0(0) 1–29

DOI: 10.1177/0735633117743918
journals.sagepub.com/home/jec
SAGE

Yune Tran[1]

Abstract

The Computer Science Teachers Association has asserted that computational thinking equips students with essential critical thinking which allows them to conceptualize, analyze, and solve more complex problems. These skills are applicable to all content area as students learn to use strategies, ideas, and technological practices more effectively as digital natives. This research examined over 200 elementary students' pre- and posttest changes in computational thinking from a 10-week coding program using adapted lessons from code.org's Blockly programming language and CSUnplugged that were delivered as part of the regular school day. Participants benefited from early access to computer science (CS) lessons with increases in computational thinking and applying coding concepts to the real world. Interviews from participants included examples of CS connections to everyday life and interdisciplinary studies at school. Thus, the study highlights the importance of leveraging CS access in diverse elementary classrooms to promote young students' computational thinking; motivation in CS topics; and the learning of essential soft-skills such as collaboration, persistence, abstraction, and creativity to succeed in today's digital world.

Keywords

computational thinking, elementary students, diverse learners

[1]George Fox University, Newberg, OR, USA

Corresponding Author:
Yune Tran, George Fox University, 414 N. Meridian St., Newberg, OR 97132, USA.
Email: ytran@georgefox.edu

Introduction

Science, technology, engineering, and math (STEM) and computer science (CS) jobs are expected to grow considerably over the next decade, faster than any other job category. Education has played a critical role in equipping this next wave of qualified and essential labor force, which is key to the U.S.' economic stability and innovation (National Academy of Sciences, 2007, 2010). Moreover, these factors have led to numerous press accounts at local and national levels to expand CS/STEM education. Most recently, President Obama in his final State of the Union (2016) addressed the crucial need renewing CS education for all students.

Given the demand to produce more qualified CS/STEM graduates, the United States has invested in improving education from these disciplines. Viable examples of such efforts included: adopting new K-12 content standards, boosting teacher quality, expanding requirements for more rigorous CS/STEM courses, improving curriculum, developing targeted professional development for teachers, promoting positive perceptions and attractiveness of CS/STEM careers, and reducing barriers for high school and undergraduate students so they may persist in CS/STEM majors. While these solutions have shown to enhance education, some critics argue that inherent problems remain. One criticism in K-12 STEM education revolved around the emphasis on the teaching and learning of science and mathematics, with less focus on technology and engineering (Bybee, 2009; Johnson & Cotterman, 2013). This lack of CS education has led to recent initiatives broadening STEM efforts to include computer programming in schools since these skills are foundational to digital technologies, problem-solving, critical thinking, and conceptualization, complimenting many academic content areas. A report by the National Research Council (NRC; 2012) suggested that teachers should develop students' understanding of computational thinking (CT) by embedding its content in existing STEM curricula given the critical role that CT plays in the 21st century (Rushkoff, 2010). Thus, the premise for this research study was to investigate over 200 elementary-aged students' CT from pre- and posttest intervention based on a 10-week curriculum that exposed them to CS interdisciplinary content learning in the classroom. The questions for the study included: (a) What changes, if any, are evident in third-grade students learning of foundational CS concepts and CT over 10 weeks of coding lessons and (b) How can 10 weeks of coding lessons influence third-grade students' CT in and out of school?

Literature Review

Background

Foundations, nonprofits, corporations, and governmental agencies are infusing monetary and human resources to fuel the CS and STEM challenge.

Past research has indicated that early exposure to STEM curricula and computing programming support positive perceptions in CS and STEM fields as well as reducing negative gender-based stereotypes (Metz, 2007). Thus, current reform efforts in the STEM movement have raised awareness for CS education integrated into existing content areas.

K-12 Education: Moving From CS to CT

With updated curricula to prepare students for college and career readiness, adopted Common Core State Standards (CCSS) and Next Generation Science Standards (NGSS) have included a more rigorous focus on critical thinking, mathematical conceptually understanding (CCSS) and CS within core scientific practices (NGSS). These standards provided a fitting backdrop for the importance of embedding CS into existing content areas as well as CT to be taught in schools. With highlights in NRC's (2012) recommendations for exposing students early to CT, Wing's (2006) seminal call, and Barr and Stephenson's (2013) notion of workforce development, the field of CS has moved to include CT into K-12 curricular frameworks. Recent scholars support propelling CS education introducing computing in K-12 instruction with specific and cross-disciplinary contexts (Israel, Pearson, Tapia, Wherfel, & Reese, 2015; Yadav, Stephenson, & Hong, 2017); thus, the initiative expands into an interdisciplinary space for teaching and learning of CS and CT.

The aforementioned research point to the importance on CS and CT integration within K-12 disciplines, particularly STEM fields, given how innovative mathematical models allow scientists and engineers to analyze, predict, and reconstruct systems that were previously impossible. Moreover, a developed framework by Sengupta, Kinnebrew, Basu, Biswas, and Clark (2013) recommended integrating CT into science and math lessons since these content areas provide a natural setting to apply algorithms, conduct scientific procedures, and design engineering structures. This priority led to computing initiatives that have been introduced and implemented across the country, although largely focused on secondary and tertiary school levels. Fewer initiatives were implemented and researched at the elementary level although some research has reinforced early exposure for elementary-aged students in CS/STEM initiatives given how these disciplines supplemented young learners' innate minds of curiosity with improved critical thinking skills (DeJarnette, 2012; Moomaw, 2012). Thus, such initiatives were crucial for raising awareness in equity issues that have persisted in the evolving technology landscape for underrepresented and underexposed students, particularly females, minority populations, English learners (ELs), and children from economically disadvantaged backgrounds (Andersen, 2005; Campbell, Denes, & Morrison, 2000).

Defining CT

Various definitions for CT have existed and emerged. An attractive viewpoint of CT has been presented by Brennan and Resnick's (2012) three-dimensional (3D) framework that included (a) computational concepts (sequencing, loops, events, and conditionals), (b) computational practices (testing, debugging, reusing, and abstracting), and (c) computational perspectives (expressing, connecting, and questioning). Terms such as computer programming and CT have been used interchangeably (Grover & Pea, 2013; Israel et al., 2015) and a recent debunking of the myth regarding CT as computer technology (Yadav et al., 2017). Early explanations of CT from the 1950s suggested it as "algorithmic thinking," a series of precise steps to solve problems and utilizing the computer when appropriate to automate the process (Denning, 2009). Other notions of the term included Wing's (2006) suggestion that CT drew on CS concepts to include understanding human behavior, problem-solving, and designing systems; a cognitive skill that an average person should possess (NRC, 2012). Bundy (2007) argued that CT was essential to every discipline given its problem-solving approach and applicability to any domain. Furthermore, these ideas were consistent and supported by the International Society for Technology in Education (ISTE) and Computer Science Teachers Association (CSTA) (see, ISTE & CSTA, 2011) to specify that CT included: formulating problems through the use of computers and various tools to solve problems; organizing and analyzing data in a logical way; representing data with visuals and models; using algorithms (a series of ordered steps) to automate solutions; identifying, analyzing, and implementing possible solutions efficiently and effectively; and generalizing problem-solving through a breadth and depth of problems. ISTE's operational definition also incorporated skills that were enhanced by vast attitudes and dispositions which were essential dimensions of CT comprising confidence in dealing with the problem, persistence in working with the problem, tolerance for ambiguity, the ability to deal with open problems, and the ability to work with others to achieve a common goal (ISTE & CSTA, 2011). CSTA's CT term included the notion that students need to learn required thinking skills to solve the most pressing problems (ISTE & CSTA, 2011).

Benefits of Early Exposure

Previous CS literature argued that elementary-aged children who participated in computer programming have improved cognitive ability, mathematics, reasoning, and problem-solving as compared with children who did not participate (Clements, Battista, & Sarama, 2001; Liao & Bright, 1991). Additionally, long-term effects for children included sustained attention, self-direction, and increased enjoyment in inquiry-based activities (Clements, 1987). Contemporary programming literature highlight the need to support innovative programming

environments and languages such as Scratch, Kodu, Etoys, and Lego We-Do that provide age-appropriate materials for children to apply core CT concepts such as abstraction, automation, analysis, decomposition, modularization, and iterative design (Bers & Horn, 2010; Mioduser, Levy, & Talis, 2009). From understanding human behaviors and solving problems efficiently to error correction and heuristic reasoning, CT should be foundational to every child's analytical ability cutting across various academic domains, reading, writing, and arithmetic skills (Wing, 2006). And despite misconceptions, children were capable of taking on these mental challenges even at the earliest grades (Bers & Horn, 2010).

Emerging practices and programming tools provided young learners an increased knowledge in computing aimed for high ceiling to challenge students' skills and low floor to allow easy access to programming (Good, 2011; Grover & Pea, 2013). Kazakoff, Sullivan, and Bers (2013) found that children's sequencing skills improved significantly from pre- to posttest after participating in a 1-week intensive robotics and programming workshop using a hybrid and developmentally appropriate tangible software called CHERP that allowed children to control the robot's behaviors through physical or tangible and graphical or on-screen movements (Bers, 2010; Horn, Crouser, & Bers, 2011).Other supporting research suggested how the learning of computing improved learners' higher order thinking skills and algorithmic problem-solving skills (Fessakis, Gouli, & Mavordi, 2013; Kafai & Burke, 2014) including enriched cognitive benefits for young gifted students who utilized Scratch to enact abstract knowledge when creating multimedia products, games, and digital storybooks (Lee, 2011).

Assessment of CT

The assessment of CT has appeared in several recent studies although a lack of research remains to comprehensively assess CT. Examples included: (a) Koh, Basawapatna, Bennett, and Repenning's (2010) assessment system based on the semantic analysis of student-created games; (b) Werner, Denner, Campe, and Kawamoto's (2012) evaluation of students' (10- to 14-year-olds) implementation of challenges in a 3D gaming environment using Alice (http://www.alice.org); (c) Brennan and Resnick's (2012) analysis of children's CT development using the 3D framework with artifacts from Scratch projects; and (d) Bers, Flannery, Kazakoff and Sullivan's (2014) evaluation of young students' (4.9- to 6.5-year-olds) written programs following activities learned from a robotics curriculum.

The aforementioned examples of CT assessments have focused on students' abilities to examine products after learning about a particular platform. These protocols presented limitations in this study given its aim to evaluate CT learning through a classroom intervention model that included concepts, practices, and perspectives occurring in a variety of tasks, contexts, and platforms. As a result, a different pre- and posttest assessment was developed. Thus, the

constructs of CT were broadly derived to Brennan and Resnick's (2012) 3D framework and ISTE and CSTA (2011) definitions with emphasis in developing basic computational concepts, problem-solving in various contexts, and reasoning about everyday scenarios. The study's pre- and posttest assessment measured development of foundational concepts as well as knowledge transfer to CT practices by making applications to functions of daily life.

Methods and Data Sources

Methods of inquiry were conducted from a sequential exploratory mixed-method approach to utilize the unique strengths of combining quantitative and qualitative data during the 2015–2016 academic year (Creswell, 2003). Data collection and analysis were emphasized through quantitative measures from students' pre- and posttest assessments while students' interviews were used for qualitative measures.

Eligible participants included all elementary-aged students from 13 third-grade classrooms enrolled in two school districts in Oregon. Control classrooms were not selected given that this type of aggressive intervention was first of its kind known at the time of study. Classrooms from the two districts were chosen for the research due to its suburban and rural settings with participants from culturally diverse and economically disadvantaged backgrounds. Informed consent was obtained through a parent permission form including access to research instruments. Appropriate Spanish translations were provided to classrooms to adhere to the schools' policies regarding correspondence written in English and Spanish to accommodate the growing Latino populations.

Total student demographics from the most recent state report card data form each district are shown in Table 1 (Oregon Department of Education, 2015)

The implementation selected all third-grade classrooms from the two districts given the equity goal for all students to receive CS lessons as well as the developmental appropriateness of concepts for that particular grade level. Moreover, recommendations were also made by Code.org affiliates for curricular implementation to this elementary-aged group (http://code.org). Undergraduate university students from a regional liberal-arts institution composed of a preservice teacher(s) and a CS student delivered the coding lessons weekly. The pairing of an education and CS student was strategic drawing on complimentary expertise in education development and computing as they worked collaboratively throughout the implementation period to plan and deliver lessons. Preservice teachers and CS students received weekly training of content and pedagogy to support students' learning of the CS integrated concepts from university faculty. Lesson planning time occurred each week and consisted of undergraduate students' discussion and practice of the concepts with faculty members offering pedagogical and differentiation instructional strategies to meet varied needs.

Table 1. Selected Student Demographics of Districts A and B.

Selected Student Demographics	Grades K–3	Grades 4–5
(a) District A 2014–2015		
Total enrolment	1,562	804
English learners	15%	19%
Economically disadvantaged	45%	45%
Students with disabilities	11%	17%
White	74%	70%
Hispanic/Latino	20%	22%
Multiracial	3%	3%
Asian	1%	2%
Black/African American	1%	1%
(b) District B 2014–2015		
Total enrolment	254	126
English learners	9%	6%
Economically disadvantaged	59%	60%
Students with disabilities	6%	17%
White	77%	79%
Hispanic/Latino	17%	13%
Multiracial	4%	5%
Asian	1%	1%
Black/African American	1%	1%

Lessons lasted about an hour each week over a 10-week period. Each lesson included approximately 30 minutes of hands-on learning of the CS concepts drawing on implications of CT and another 20 minutes of online applications using adaptations from CSUplugged (Bell, Witten, & Fellows, 2011) and Course 2 of the elementary framework that used Blockly programming language from Code.org (http://code.org). These programming tools were purposefully selected for the hybrid lessons to integrate hands-on learning of concepts and an online media rich programming environment. Both tools aimed at engaging elementary-aged students with accessible and entry points to the concepts (low ceiling) but challenge their mental capacities with more difficult lessons (high ceiling; Good, 2011; Grover & Pea, 2013). Furthermore, elementary students were paired with partners throughout the 10-week period as they engaged in hands-on practice of the concepts, discussed its relationship to core subjects both in and out of school, and interacted with the visual online tools.

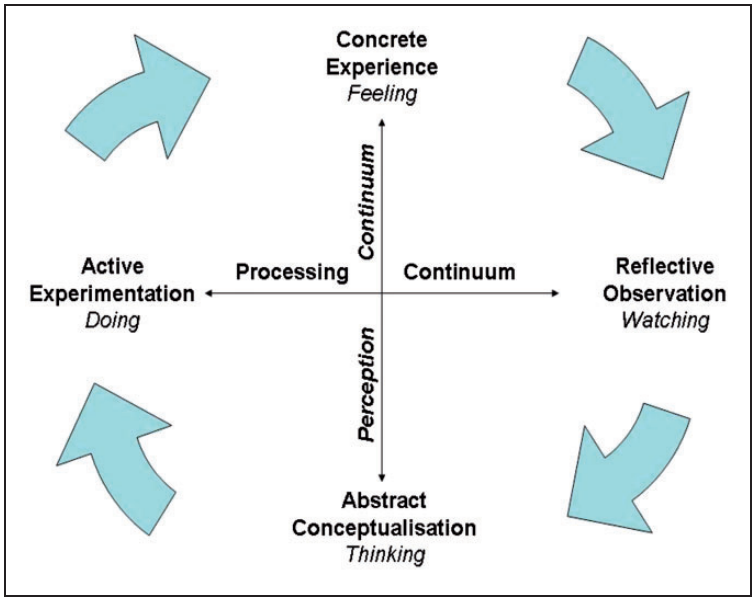


Figure 1. Model of Kolb's theory.

The intervention framework was derived from Kolb's (1984, 1999) theoretical model (see Figure 1) from which experience is central to learning and development.

Kolb (1984) suggested that students practice four parts of the model when learning new concepts to ensure the best learning and content transfer. For instance, hybrid off-line or online activities around the concept of algorithm involved concrete experience with graph paper and symbols to write the algorithm and relate concepts to prior knowledge. Reflective observation involved thinking about how concepts were presented with hands-on materials and discussing how those steps were performed in real life. Students reflected on the steps of algorithms on graph paper and dialogued the critical steps to related activities such as riding a bike or making a peanut-butter-jelly sandwich. Abstract conceptualization involved internalizing the learning, acting on the concept, and considering where to move blocks to execute a program with the online puzzles. Active experimentation involved students jumping in, exploring, and doing the online puzzles.

For the model to be effective, experiences should be leveraged and accessible through active learning (Freeman et al., 2014) with rich interdisciplinary content and context-embedded frameworks that combine CT with science (i.e., trial and error with experiments linked to debugging), CT with math (i.e., writing down step-by-step operations from a story problem using algorithm or minimizing

steps in a math problem using the concept of loops), and CT with language arts (i.e., storytelling and song writing using the concept of function). Thus, the goal of the intervention was to introduce foundational CS concepts by having students practice a variety of coding activities through lessons that enhance CT with discussion of relatable content and real-life applications learned from different subjects or home experiences.

Elementary students completed a pre- and posttest on CT that was developed by the researcher in consultation with the university's CS department. Constructs of the assessment included 10 written items that measured different computational concepts, practices, and perspectives with two questions per concept related to sequence, algorithm, looping, debugging, and conditionals (see Appendix A). The researcher pilot tested the instrument to a group of third-grade students unrelated the research project to ensure appropriate vocabulary and understanding of the constructs. An example of a sequencing test item included: "Put these mixed-up instructions for baking a cake in order using only four steps. Write numbers 1-4 next to those steps." In this task, students analyzed each step of the algorithm drawing on CT practices and perspectives by relying on familiar experiences. Students completed the instrument by hand in class before and after the intervention period. Accommodations were provided to students who needed oral administration, extra time, or text translation. Assessments were collected and scored for each item (1 = *correct* and 0 = *incorrect*) following each administration. In addition to the assessments, an average of 4 students per classroom, or 52 students overall with parental consent, participated in several random semistructured interviews during the 10-week lesson cycle. At the start of each interview, the researcher initiated conversations about students' enjoyment of interests, tasks, and fun activities to build students' trust, comfort, and confidence in answering questions. Once students were at ease, the researcher posed a series of questions related to learning outcomes of coding lessons and students' perceptions of in and out of school-related activities. Rarely, did the researcher have to prime or prompt students for answers. The majority of students shared eagerly and reflected positively on their learning experiences since the topics were relatively new to them. Examples of questions included: (a) What are you learning from coding lessons when Mrs. W. or Mr. X. comes to your classroom each week?, (b) How do you feel about these lessons?, and (c) Is coding important in life? Interviews were recorded and transcribed to produce descriptive codes and themes.

Interviews utilized theme analysis with techniques that searched for word repetitions or key words in context (Strauss, 1992), a careful reading of larger blocks of text to compare and contrast (Glaser & Strauss, 1967), and an intentional search of linguistic terms (i.e., because, so, and so on) that described causal relationships (Strauss & Quinn, 1997). Thus, the methods of inquiry were conducted from an exploratory mixed-method approach to utilize the beneficial strengths of quantitative and qualitative research paradigms (Creswell,

2003; Patton, 2002). Results report quantitative measures of the survey instrument using (STATA) software. Other major qualitative themes from students' responses on the interviews were related to (a) specific recall of learning concepts from CS lessons; (b) relationship of learned CS concepts to everyday life; and (c) essential development of soft-skills such as collaboration, teamwork, and persistence. Utilizing the definition term of CT drawn from Wing's (2006) perspective, students made connections of concepts learned in the recognition of key CT ideas associating them with everyday life citing examples of soft-skills and linking attitudes and dispositions to essential CT domains (ISTE & CSTA, 2011).

Findings

Results From CT Assessment

Results from elementary students pre- and posttest assessments included 263 pretests and 244 posttests from 13 teachers in five schools. Students identified themselves by first name only. Of all identified individuals, 183 could be matched on the two tests. Scores ranged from 0 to 10 and internal reliability coefficients were .63 on the pretest and .61 on the posttest. These figures are low but expected of an exploratory mixed-method study. The instrument will be revised to achieve higher measures in future administration. The percentages correct increased for all teachers and the changes were significant for five teachers ($p < .0038$, with a Bonferroni correction for familywise error) and the overall increase from pre- to posttest was significant, $t(182) = 9.62$, $p < .0001$. The second loop question (Item 6) and the first debugging question (Item 7) did not make significant gains although there was an attempt to create assessment items with language couched in normal everyday usage. A two-step process for those questions likely presented confusion for students. The loop question required students to circle the repeating patterns and then rewrite them only once on the line, whereas the debugging question required students to cross out incorrect steps and then rewrite them correctly on the line. Thus, one-step procedures are best for reducing technical jargon and will be corrected on future assessments. Table 2 represents the percentage increases and change for each test item from pre- and posttest, whereas Figure 2 shows the same information in the form of a line graph. It appears that the activities implemented under this type of intervention where lessons included hands-on activities, discussion, and online application are generally successful in promoting CS understanding for young students.

Results From Student Interviews

Student interviews produced particular examples of specific concepts that were acquired from the lessons and how those concepts influenced tasks both in and

Table 2. Pre- and Posttest Percentages, Correct by Item, Total Score.

Assessment construct	Item	Pre	*SD*	Post	*SD*	Total	*SD*	Change
Sequence	1	0.51	0.50	0.74	0.44	0.62	0.49	0.23
Sequence	2	0.45	0.50	0.62	0.49	0.54	0.50	0.17
Algorithm	3	0.61	0.49	0.80	0.40	0.70	0.46	0.19
Algorithm	4	0.28	0.45	0.40	0.49	0.34	0.47	0.11
Loop	5	0.21	0.41	0.39	0.49	0.30	0.46	0.18
Loop	6	0.03	0.18	0.08	0.27	0.05	0.23	0.04
Debug	7	0.09	0.29	0.08	0.27	0.08	0.28	−0.02
Debug	8	0.45	0.50	0.69	0.46	0.57	0.50	0.23
Conditional	9	0.58	0.49	0.70	0.46	0.64	0.48	0.12
Conditional	10	0.11	0.32	0.22	0.42	0.17	0.38	0.11

Note. Overall increase from pre- to posttest was significant at $p < .0001$ level.

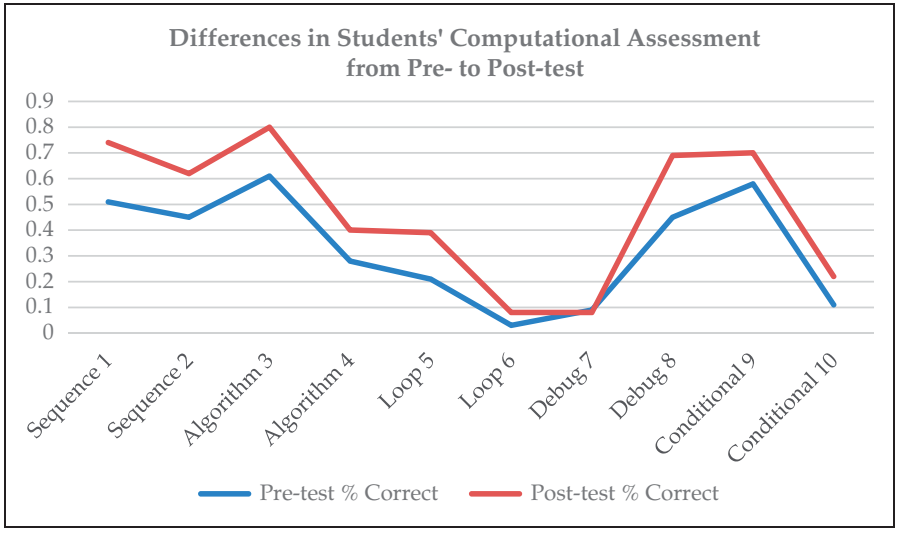


Figure 2. Line graph representation of computational assessment.

out of school. Nearly half of the 52 students (40%) who participated in the interviews were able to provide specific concept that they learned: 8 students recalled the concept of loops, 7 recalled the concept of debugging, and 6 recalled the concept of algorithm as learning milestones. Other students made references

to the enjoyment of lessons and the opportunity to learn something different as well as the ability to play on the i-pads. Thus, an overwhelmingly 96% of students (50 of 52) had positive reactions to the lessons saying words such as: ''excited, really good, happy, it's fun, I'm learning new stuff, I look forward to the lessons, and I'm enjoying them.'' The responses of males resorted to short one or two words whereas that of females were more elaborate in how they felt about the lessons. The remaining two students were neutral in attitude about how they felt about the lessons although they did specify that they would prefer more challenging lessons.

Specific learning experiences from coding lessons. The various concepts of loops, debugging, and algorithm with their definitional uses were prevalent in the majority of students' learning responses. Participants shared understandings of these concepts highlighting examples that were tied to hands-on and online lessons. Tables 3 to 5 depict the three major concepts and learning responses by students including their gender, age, ethnicity, and EL background. Rather than individual economic descriptions for the cases participating in the interviews, the total percentages for economically disadvantaged for all students participating are included using criteria from the district's free and reduced lunch program and special education qualification.

Relationship of CT to essential everyday life. Students explained the connections of lesson concepts as they related to important functions in everyday life. Selected key principles in coding lessons promoted students' understandings and applications within the environment. Several responded as to how the lessons associated to understanding the world with these comments:

- ''Your morning routine. You wake up, get dressed, brush your teeth, and brush your hair. Well actually, you wake up, get dressed, brush your hair, eat breakfast, and then brush your teeth. (Betty)
- ''It's helpful because later if your computer doesn't work then you know how to fix it.'' (Gigi)
- ''You can fix your mistakes and it will help you get successful.'' (Kelly)
- ''It's when you say, go touch the door, and come back, go touch the door, and come back, and go touch the door, and come back. Instead of saying it 3 times just say, Hey Ethan, go touch the door three times.'' (Zeus)
- ''The best part is like when we get to do the online games and talk about things. Like math is pretty important, because those that make people smarter they start out by using math.'' (Mike)
- Learning about coding will help me in a future job. I might want to be a judge and a judge needs to use teamwork by communicating with their helpers and the people.

Table 3. Students' Background Information and Learning Responses Related to Loops.

Student (pseudonyms)	Learning response	Background
Adolfo	During coding, I learn about looping. Looping means like instead of saying, "move forward, move forward, move forward" a bunch of times you can just loop it and put it in the little squares and things. And for the game you can choose a number for how many times you want to do it.	Latino male, age 9 years, EL student
Amy	About looping. Repeating things like, if someone tells you, "go touch the door and then go back to your seat" and then they tell you to do it again and again, they could just say, "go touch the door and go back to your seat three times.	White female, age 9 years
Ben	We are learning to make this little person from other games Basically to make him do it over and over again. And I can use just a few codes. So I can type in number 5 and I put something in the code block, the repeat block and he would do that 5 times. So if I made a pattern and he did it 5 times, so if it was diagonal like this and then here's where he wants to go, I could say, "Forward, forward, turn right or left, and then forward, forward" and so on. And then I could count like the this . . . four, five, six and if there were six I would put in the number six and it would go like this 6 times.	White male, age 8 years
Jake	I'm actually learning that computer coding when I do it its actually getting better and better every day & I'm enjoying it. I'm learning looping (Let's say you need to sharpen 10 pencils, you don't say "sharpen the pencil, (says that x10 in mind). All you have to say is sharpen the pencil x ten. It helps in life. Algorithm it's like a series. Let's say you want to make a fortress so you need to set up at least four chairs, some blankets & I remember you need steps.	White male, age 8 years
Jane	Right now we are learning about looping. Looping is when easier to repeat stuff and you don't have to say it over and over again. You can x it with whatever you want to put & x it.	White female, age 8

(continued)

Table 3. Continued

Student (pseudonyms)	Learning response	Background
Niko	A loop is where you have something, like an algorithm, here is the zombie and then there's the path and there's the chomper plants that you have to get him to here, you have to get him to the sunflower. So you pick an algorithm, the loop button is kind of shaped like that, and it lets you pick what you want to do. You pick your thing and put it inside there. Like the normal pattern, without using the loop button, would be, move forward, move forward, move forward, turn left, move forward, move forward, move forward, but with that, you could do "move forward x3" and then you would do "turn left" then "forward x3" and that's how you use the loop.	Asian male, age 8 years, EL student
Quincy	It's really fun. I feel good. They are helping like shortening things. Sometimes I use looping for other things like for writing. Instead of saying and a dog and a cat, I say it repeats.	Latino female, age 8 years, EL student
Mike	Loops are something you can do over and over again.	White male, age 8 years

Note. EL = English learners.

Table 4. Students' Background Information and Learning Responses Related to Debugging.

Student (pseudonyms)	Learning response	Background
Andy	Debugging like, we can fix the mistake. We were having a relay race and we were running to the paper and if the person in front of you did something wrong you would circle it and put a line through it and then put the correct thing, the correct arrow.	White male, age 9 years
Cameron	Debugging is like we're playing a game right now and we have to tell the bee how to get to his flower to make honey or get to a honey comb to make honey. And if it's supposed to go left and if we accidently tell it to go right we have to debug and make it a turn that you want it to go.	African American male, age 8 years
Gigi	I'm learning new words. I'm learning how to program. That most of this stuff is part of our everyday lives. Debug is like to correct when sometimes someone else is correcting some test, and you get the answer wrong, and they write down the right answer that is debugging.	Latina female, age 8 years, EL student
Paul	The best thing about the coding class was I liked debugging because xxx was making mistakes and I fixed his mistakes.	White male, age 9 years
Beau	The one that we're doing right now, where we're fixing and debugging. It's where we have to fix something. Sometimes when people put in wrong steps, you have to fix them	White male, age 8 years

Note. EL = English learners.

Table 5. Students' Background Information and Learning Responses Related to Algorithm.

Student (pseudonyms)	Learning response	Background
Joe	Program is like you're doing by following an algorithm. And an algorithm is a list of instruction.	White female, age 9 years
Macy	Like algorithm, where there is a process. Like steps, like building a plane or to brush your teeth is in steps.	African American female, age 8 years, EL student
Maria	The word algorithm is a list of things that you do. An algorithm we are doing a dance thing & it's like we are doing clap, clap, clap 3x, its hands on head, hands on hips, so then at the end we do spin. So you do clap again, clap 3x and hands on head, hands on hips repeat. Clapping 3x is the algorithm	Latina female, age 8 years, EL student
Eve	Algorithm is kind of like ordering stuff. Like from one to ten like library. Drawing books only go to the area of drawing books. Like if there are twenty books of drawing you will put them in order.	White female, age 9 years
Ricardo	In math you use algorithms, like when you do an equation	Latino male, age 9 years, EL student
Mike	Algorithms are like steps, like to make a sandwich.	White male, age 8 years

Note. EL = English learners.

These students' responses revealed a construction of relevant examples to explain the interconnectedness of CT to daily life. Through problem-solving and critical thinking, students were able to make connections that link CS concepts to their own world.

Essential soft-skills such as teamwork and persistence are critical in life. Students identified social benefits in the coding lessons, and that, the lessons advanced skills in teamwork and collaboration between peers. Some students referenced how the lessons improved their abilities to share and balance turn taking. The responses following indicate this notion.

- "I feel good about them. I like the part where we get to be with our friends. It's teaching me how to take turns and stuff. Last year I wasn't doing a very good job at taking turns." (Abel)
- "It actually is helping me with teamwork which I'm not good at." (Carla)
- "Learning about coding will help me in a future job. I might want to be a judge and a judge needs to use teamwork by communicating with their helpers and the people." (Jen)
- "My favorite part is probably working with my partner to help the bee find its nectar and honey. It's teaching me how to be more friendly and be nice to others." (Betty)
- "I learned in coding working together. Then I will be a lifeguard because it is working together like a lifeguard saves the drowner and another lifeguard keeps the people away." (Ted)
- "I think that it will help me with partnering up because in the SWAT team people pair up and work together." (Tim)
- "Buddy working will help me in a future job. I know this because if you were going to be a vet you would have to work with the other vets. "(Vicky)
- "If you are an engineer, teamwork will be one important thing in this job." (Luke)
- "Learning about coding will help in a future job because working together with a partner will give me practice for the future." (Quincy)
- "When you are a tailor, if more than one person is helping, teamwork will be very useful to help communicate." (Rosie)
- "It teaches you teamwork with your partner. Sometimes they can be challenging and sometimes can be easy." (Kyle)
- "I'm learning that you should share with your partner and that sharing you should always be nice, don't be a bossy navigator or a bossy driver. It's important for later in anything." (Jane)

The notion of persistence and working hard to not give up was a common skill identified by students as a valuable tool to be used later in life. This ability to endure and persist through the new and challenging content enabled students

to continue problem-solving within the online puzzles to correct errors from programs. Not only were students motivated to complete the level to move onto the next, but also they acquired a sense of work ethic that allowed them to continue the challenging task. Several examples that emphasized this point were:

- You guys inspired me that not everything is perfect but you still have to try hard and use teamwork. You need help. Teams help a lot.'' (Joe)
- ''Coding is really, really fun! And it's really hard! But, you never give up, because you know you will get on a higher level if you don't give up. At first it seems really hard, but so on it's like ''Hey! I know this and it's really easy!'' But at first you think that you won't be able to do it and so on. Then you're like, ''Hey! I can do this; it is really easy.'' (Harriet)

These examples demonstrated that when children try something new and different, they are easily frustrated; however, for Harriet and Joe, they were able to complete the task regardless of the difficulty. Qualities such as having patience, getting practice, and giving your best effort were significant attributes that allowed them to achieve tasks. And despite the challenges they encountered, students were able to work collaboratively with partners to persevere through the puzzles with the correct code to move the character in the game.

Discussion

This study sought to answer the two research questions related to an implementation of CS coding lessons provided during in-school time to elementary-aged students who were attending increasingly diverse suburban and rural schools. In doing so, it became apparent that the findings about students' learning in computing supported previous literature on the advantages reaped from early exposure of certain CT activities as well as leveraging equity to allow all students a chance to learn CS.

In answering the first research question, it became evident that students' understanding of specific CS concepts particularly related to algorithm, loops, and debugging improved from pre- and posttests based on the intervention. Extensive research has focused around issues of what constitutes as CT across K-12 education (Barr & Stephenson, 2011); the teaching and learning of CS skills, concepts, and practices (Grover & Pea, 2013); and the environments on how CS skills are learned (Brennan & Resnick, 2012; Cooper & Cunningham, 2010). One approach of CT presented within the CS education community involved a list of concepts mapped according to content disciplines such as the algorithm used in in science, how abstraction unfolds in social studies, or automation in math (Barr & Stephenson, 2011). The other approach previously mentioned provided operationalizing CT in K-12 education in terms of the 3D model with articulation in how to engage students for each level with its diverse uses (Brennan & Resnick, 2012). Yet, current research continues investigation of

the creation of frameworks for understanding and evaluating CT as well as the implications of learning environments, tools, and activities that promote such learning (Weintrop et al., 2014). Important to note in this study was the use of varied environments in designing the intervention program that drew upon adapted lessons from CSUnplugged (Bell et al., 2011) and the Code.org (http://code.org) framework. The intersections of the CT environments were derived from three broad categories of computer programming consisting of (a) simulation and modeling (Basawapatna, Repenning, Koh, & Savignano, 2014), games design (Reppening, Webb, & Ioannidou, 2010), and visual or tangible blocks such as Scratch (Resnick et al., 2009). The three environments were practiced during the intervention when students (a) learned the concept of algorithm using a simulation activity modeled by the preservice teacher and CS student to carry out the steps to make a sandwich, (b) matched blocks with the appropriate event handler to create a game using the event handler blocks ((http://code.org; Course 2, Flappy Lesson), and (c) used loops to write a program for employing a combination of commands to make the bee perform the actions on the computer.

Furthermore, students employed a variety of simulation exercises to gain knowledge of CS concepts while transferring that knowledge to enhance CT development. For the concept of debugging, students engaged in a relay race where team members collaboratively edited errors in a program where as in the concept of loops, students performed loops through a repeated dance activity and created new repetitive motions using pop songs. Students had opportunities for discussion of the concepts before transitioning to apply understandings to the online platform to drag, drop, blocks for game design, or created new tasks to program the characters from Code.org. Research has suggested that computational rich environments and effective tools should encourage "low floor and high ceiling" principles to allow novice learners to create programs (low floor) with sufficient content to satisfy advanced programmers (high ceiling; Grover & Pea, 2013). The hybrid environment with curricular hands-on activities were ideal methods to motivate and scaffold elementary-aged students' understanding of introductory CS concepts, thus, promoting CT and enabling transfer of the content by utilizing the programming language embedded (Resnick et al., 2009) with the visual or graphic blocks from Code.org. The lessons were systematic and enabled transfer while advancing equity within CT platforms (Reppening et al., 2010) given their delivery during the day as a formative experience inclusive of all learning backgrounds. Moreover, the intervention program was delivered to selected underrepresented elementary-aged students enrolled in the two districts with economically disadvantaged backgrounds (between 45% and 59%), a growing Latino population (between 17% and 20%); ELs (between 9% and 15%), and special education (between 6% and 11%). This type of learning might not have been possible with the lack of teacher knowledge and resources from the two districts.

In answering Research Question 2, the benefits of early exposure promoted students' positive attitudes about CS learning, enriched CT and problem-solving skills, and generated interest about CS in everyday life. Basic math and science literacy have been fundamental goals for all children; however, additional computer competencies have fostered analytical and problem-solving abilities needed in schools (NRC, 2012). Participants made crucial connections to out-of-school activities including on-demand job-related soft-skills such as communication and teamwork, highlighting benefits gained from the opportunity to learn coding. The legacy of unique thinking skills that children developed with CT has shared elements to mathematical, engineering, and design (Lee et al., 2011). Moreover, international curriculum statements have supported essential critical thinking skills that young students need to progress in the 21st century with necessary competencies for a successful future (Australian Curriculum Assessment & Reporting Authority, 2010; Organisation for Economic Co-operation and Development, 2015). Particular higher order thinking skills have been traditionally emphasized through analysis, synthesis, and evaluation taught in subject areas (Bloom, 1956) as well as solving ill-structured problems by observing, inferring, and predicting through processes such as interpreting data, manipulating variables, and developing solutions (Lewis & Smith, 1993). Students in this study developed CT as they actively discussed problems related to debugging, loops, and algorithms by finding optimal solutions for the unplugged activities and online puzzles; thus, challenging each other's thinking as they discovered different ways to solve complex problems or make associations of those concepts to the outside world. Previous research has highlighted how Scratch programming improved effects of logical reasoning and problem-solving skills in young learners (Lee, 2011) including cross-curricular benefits such as the importance of computing skills that are used to assist students to read, understand, and adapt algorithms in various situations (Brown & Kolling, 2012). Finally, students enacted CT skills using various methods on the final lesson that involved animated creations using Code.org characters for interactive art for digital stories (Calder, 2010).

Most interesting from the study was the innate ability of students to persevere as they engaged in collaborative work rather than expecting immediate success when confronted with challenging tasks. The learning experiences made indelible impact on participants as they embraced collaborative challenges to persist mentally and academically. Increased computational competencies primed early interest for participants to engage in future CS studies, thus, promoting positive perceptions as they enter secondary school or beyond. Utilizing pair programming promoted social skills as mentioned in previous studies (Lewis, 2011; Scaffidi & Chambers, 2012) that were significant in advancing students' attitudes, understanding, and interest in CS. Not only were students engaged and motivated by the tasks, they also learned to navigate cooperatively by asking questions about unplugged activities while employing reasoning techniques to discover various patterns that were embedded in the online puzzles and games.

This studied investigated the impact of CT opportunities provided to young children and found how students made connections to curricular subjects. It may be worth noting that CT practices have been integrated across a variety of subjects and findings (Sengupta et al., 2013) and this study confirmed that early learners were capable of making associations. As noted from the literature and consistent with research (Fessakis et al., 2013; Papert, 1991), the power of computing increased students' problem-solving and higher order thinking skills as they learned the embedded content and practiced them through interactive visualization from Code.org. Embedding CT practices into subjects such as history, language arts, mathematics, and science have been supported (NRC, 2012; Sengupta et al., 2013; Yadav et al., 2017). Some research has specified bringing CT into K-12 classrooms with STEM learning through online computational resources such as the iLab Network to enable students to use instruments and interact with scientific phenomena by controlling experimental equipment and taking measurements of the material in real time (Jona et al., 2014). In this way, students can make deeper connections of CT practices with science and technology in an engaging and meaningful way. Regardless of the medium, advocates of CT agreed that the urgency to teach and strengthen CT as a priority for K-12 classrooms. This study highlighted one example of how CT allowed elementary-aged students from diverse backgrounds to thrive in soft-skills, critical thought, reasoning, and understanding.

Implications and Future Directions

CT has gained momentum in schools with considerably press and resources in schools to provide American students with fundamental skills needed to compete in the global economy. This study attempted to document the effects from an aggressive intervention that leveraged opportunities for learning during the day. Future studies must be sensitive to the processes that occur at micro and macro level for implementing the growing demand of CT in schools. Barriers such as access to technology, resources, time, student demographics, environmental conditions, preservice and inservice teacher knowledge, and professional development are associated with implementation. Additional research is needed to investigate the different types of software, programs, courses, or platforms used in elementary instructional contexts as well as how they are ensuring equity for participation or reducing barriers effectively.

Teacher content knowledge, pedagogical practices, and instructional strategies are important factors to explore in future research. Increased professional development opportunities provided from code.org including federal, state, and local initiatives have broadened participation for inservice teachers to deliver CT in classrooms across the country. More of these types of learning experiences are needed for inservice teachers to support students' understanding of CT concepts including application to the disciplinary knowledge related to specific subject

area. Inservice teachers require professional development on best practices to possess skills and confidence to prepare engaging lessons and deliver meaningful. Finally, inservice teachers need training on appropriate hybrid forms of learning which include physical and digital tools in the curriculum so that students are exposed to computational practices in core and interdisciplinary content.

A complimentary approach of teacher knowledge and applicable tools exist for preservice teachers as well. Providing preservice teachers with the requisite courses in content, pedagogy, and instructional practice necessitates preparation as they work with K-12 students in the digital world. Embedding this type of background knowledge, skills, and appropriate framework for teaching core content with interdisciplinary CT is fundamental in any preparation program regardless of licensure level or content specialization. Updated CT preparation can help to move the pendulum so that preservice teachers transfer their knowledge and capacity to influence learning in their future classrooms. In this way, CT moves from an interdisciplinary space to a transdisciplinary environment so students master critically thinking to succeed across multiple fields of study (i.e., journalism, finance, physics, language arts, or biology). Finally, education and CS faculty in K-16 platforms should collaborate given complimentary expertise on the best ways to support and improve the teaching and learning of CT.

Given the elements of soft-skills that are essential traits that can be taught and learned, elementary-aged students could benefit from lessons that embed these traits into the standard curriculum. Fictional stories and expository texts that highlight characters with these traits could be ways to naturally discuss them in class. Arranging time for these discussions can be explicit with character analysis during small group reading or within the larger class as students and teachers share personal stories about difficult challenges while proposing strategies for success. Elementary-aged students should have conversations with their peers and teachers around similar characteristics with some common sayings such as: ''I can do it.'' ''I won't quit.'' ''It's hard the first time, but it will get easier.'' These conversations and discussions are helpful to students as they problem solve with grace and ingenuity when overcoming obstacles and mistakes in class or on homework. Equally important in nurturing these soft-skills would be for teachers to resist the temptation of giving students the answer on certain tasks as they face challenges. Ideal environments include inquiry-based lessons and practices that allow students to collaborate with peers, construct arguments, investigate theories, apply thinking, and assimilate new knowledge. Thus, teachers need to develop content knowledge, skills, and practices that integrate digital and physical components to expose and enrich students' critical thinking capacity. In this way, students practice and use CT to find solutions to complex problems, persevere, and build confidence as they complete tasks related to the real world since these skills are vital to their achievement within the global market due to the rapid changes of technology.

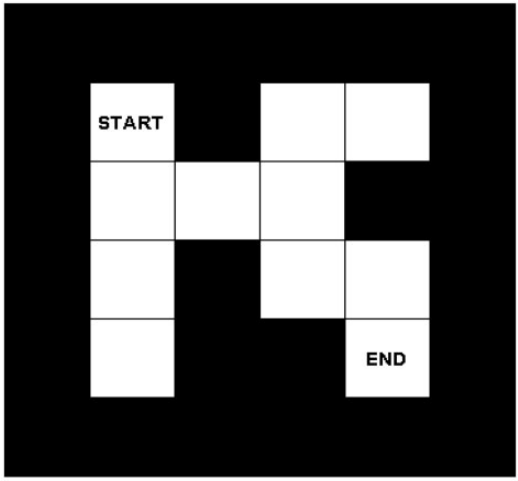
Finally, a future direction includes revision to the assessment instrument. A revised assessment will be administered to experiment groups to measure students' CT skills. Given past results from the project, future studies may also involve second- or third-level effects, with investigators collecting data on implementation and context that might explain higher level factors. For instance, three of the teachers with nonsignificant results came from the same school, suggesting there may have been a systematic difference in implementation at that site. Future use of the assessment would include revision of wording of all items as well as stronger alignment with curricular content.

Conclusion

The current technology landscape demands students who are prepared with related CS or CT skills for growing workforce needs. It has been documented that integrated CT is imperative in K-12 education. This study supported that priority as well as extending CS opportunities to all students from diverse background experiences, especially those most vulnerable who have lacked resources to develop critical CT skills and applications to in- or out-of-school experiences. In summary, this study initiated the call of leveling the playing field in CS education within the elementary space so that all students, regardless of gender, race, language, economic status, or intellectual ability, can emerge as a CT winner and succeed in the 21st century.

Appendix A

Computational Thinking Assessment



1. Using the words "up," "down," "left," and "right," write down the instructions for completing the maze (e.g., up, up, left, down).

_____ _____ _____ _____ _____
_____

2. Now, write instructions for completing the maze going backwards (starting at "END" and getting to "START").

_____ _____ _____ _____ _____
_____

3. Put these mixed-up instructions in order by writing numbers 1–6 next to each step.
   - Pick the ripe tomatoes. _____
   - Plant seeds. _____
   - Buy seeds. _____
   - Eat the tomatoes. _____
   - Watch the plants grow. _____
   - Water the seeds. _____
4. Put these mixed-up instructions for baking a cake in order using only four steps. Write numbers 1 to 4 next to those steps.
   - Make a salad. _____
   - Pour batter into pan. _____
   - Eat half of the batter. _____
   - Mix ingredients in a bowl. _____
   - Drink some water. _____
   - Bake for 20 minutes. _____
   - Measure ingredients. _____
5. Emma is exercising before gym class. Emma does two push-ups. Emma repeats the first step three times, and touches her toes once after each repeat. How many push-ups did Emma do? _____

   How many times did she touch her toes? _____

6. Rewrite these instructions so that loops (repeating patterns) are only written once.
   - Do three pull-ups.
   - Do three pull-ups.
   - Drink water.
   - Do three pull-ups.
   - Do three pull-ups.
   - Drink water.

- Do three pull-ups.
- Do three pull-ups.
- Drink water.
- Have a snack.

7. Circle the wrong steps in the sequence.
   - Wake up.
   - Get dressed and eat breakfast.
   - Drive to school.
   - Put on your backpack for school.
   - Get in the car.
   - Walk into the classroom.
8. Rewrite the sequence from above to make it correct.
9. If Emily has three or more apples, she can make an apple pie. Emily has five apples and one pumpkin. Can she make an apple pie?

   Circle Yes or No.

10. If John has two bananas, he will share one with his friend. Else, John will keep the banana. John has one banana and two carrots. What is John going to do?

Declaration of Conflicting Interests

The author declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author received no financial support for the research, authorship, and/or publication of this article.

References

Andersen, M. (2005). Thinking about women: A quarter century's view. *Gender and Society*, *19*(4), 437–455.

Australian Curriculum Assessment & Reporting Authority. (2010). *Digital technologies in the Australian curriculum*. Retrieved from http://www.australiancurriculum.edu.au/

Barr, V., & Stephenson, C. (2011). Bringing computational thinking to K-12: What is involved and what is the role of the computer science community? *ACM Inroads*, *2*(1), 48–54.

Basawapatna, A. R., Repenning, A., Koh, K. H., & Savignano, M. (2014, March). The consume-create spectrum: Balancing convenience and computational thinking in STEM learning. In *Proceedings of SIGCSE 2014 conference* (pp. 658–664). New York, NY: ACM.

Bell, T., Witten, I., & Fellows, M. (2011). *Computer science unplugged*. Retrieved from http://csunplugged.org/books/

Bers, M. U. (2010). The TangibleK robotics program: Applied computational thinking for young children. *Early Childhood Research & Practice, 12*(2). Retrieved from http://ecrp.uiuc.edu/v12n2/bers.html

Bers, M., & Horn, M. (2010). Tangible programming in early childhood: Revisiting developmental assumptions through new technologies. In I. Berson & M. Berson (Eds.), *High-tech tots: Childhood in a digital world* (pp. 49–70). Charlotte, NC: Information Age Publishing.

Bers, M. U., Flannery, L., Kazakoff, E. R., & Sullivan, A. (2014). Computational thinking and tinkering: Exploration of an early childhood robotics curriculum. *Computers & Education*, *72*, 145–157.

Bloom, B. (1956). *Taxonomy of educational objectives book 1: Cognitive domain*. New York, NY: Longman.

Brennan, K., & Resnick, M. (2012, April). *New frameworks for studying and assessing the development of computational thinking*. Paper presented at the Annual Meeting of the American Educational Research Association, Vancouver, Canada.

Brown, N., & Kolling, M. (2012, July). Position paper: Programming can deepen understanding across disciplines [DRAFT]. In *IFIP working conference-addressing educational challenges: The role of ICT*. Manchester, England: Manchester Metropolitan University. Retrieved from http://www.cs.kent.ac.uk/people/staff/nccb/position_paper.pdf

Bundy, A. (2007). Computational thinking is pervasive. *Journal of Scientific and Practical Computing 1*, *2*, 67–69.

Bybee, R. W. (2009). K-12 engineering education standards: Opportunities and barriers. *Workshop on Standards for K-12 Engineering Education*. Washington, DC: National Academies Press.

Calder, N. (2010). Using Scratch: An integrated problem-solving approach to mathematical thinking. *Australian Primary Mathematics Classroom*, *15*(4), 9–14.

Campbell, G., Denes, R., & Morrison, C. (Eds.). (2000). *Access denied. Race, ethnicity, and the scientific enterprise*. Oxford, England: Oxford University Press.

Common Core State Standards http://www.corestandards.org/ (accessed 15 November 2017)

Clements, D. H. (1987). Longitudinal study of the effects of logo programming on cognitive abilities and achievement. *Journal of Educational Computing Research*, *3*, 73–94.

Clements, D. H., Battista, M.T., & Sarama, J. (2001). Logo and geometry. *Journal for Research in Mathematics Education. Monograph*, 10, I–177.

Cooper, S., & Cunningham, S. (2010). Teaching computer science in context. *ACM Inroads*, *1*, 5–8.

Creswell, J. W. (2003). *Research design: Qualitative, quantitative, and mixed methods approaches* (2nd ed.). Thousand Oaks, CA: Sage.

CSTA. (2011). *Operational definition of computational thinking for K-12 education*. Retrieved from http://www.csta.acm.org/Curriculum/sub/CompThinking.html

DeJarnette, N. K. (2012). America's children: Providing early exposure to STEM (science, technology, engineering and math) initiatives. *Education*, *133*(1), 77–84.

Denning, P. J. (2009). The profession of IT beyond computational thinking. *Communications of ACM*, *52*(6), 28–30.

Fessakis, G., Gouli, E., & Mavroudi, E. (2013). Problem solving by 5-6 years old kindergarten children in a computer programming environment: A case study. *Computers & Education*, *63*, 87–97.

Freeman, S., Eddy, S. L., McDonough, M., Smith, M. K., Okoroafor, N., Jordt, H., ... Wenderoth, M. P. (2014). Active learning increases student performance in science, engineering, and mathematics. *Proceedings of the National Academy of Sciences*, *111*, 8410–8415.

Glaser, B. G., & Strauss, A. (1967). *The discovery of grounded theory: Strategies for qualitative research*. New York, NY: Aldine.

Good, J. (2011). Learners at the wheel: Novice programming environments come to age. *International Journal of People-Oriented Programming*, *1*(1), 1–24.

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

Horn, M. S., Crouser, R. J., & Bers, M. U. (2011). Tangible interaction and learning: The case for hybrid approach. *Personal and Ubiquitous Computing*, *16*(4), 379–389.

International Society for Technology in Education & the Computer Science Teachers Association. (2011). *Operational definition of computational thinking for K-12*. Retrieved from http://www.iste.org/docs/ct-documents/computational-thinking-operational-definition-flyer.pdf?sfvrsn=2

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.

Johnson, H., & Cotterman, M. (2013, November). *Collaborative efforts to put the 'E' back in STEM* (p. 3). Arlington, VA: NSTA.

Jona, K., Wilensky, U., Trouille, L., Horn, M. S., Orton, K., Weintrop, D., & Beheshti, E. (2014). *Embedding computational thinking in science, technology, engineering, and math (CT-STEM)*. Presented at the Future Directions in Computer Science Education Summit Meeting, Orlando, FL.

Kafai, Y.B., & Burke, Q. (2014). *Connected Code: Why children Need to Learn Programming*. MIT Press.

Kazakoff, E., Sullivan, A., & Bers, M. U. (2013). The effect of a classroom-based intensive robotics and Programming workshop on sequencing ability in early childhood. *Early Childhood Education Journal*, *41*(4), 245–255.

Koh, K. H., Basawapatna, A., Bennett, V. & Reppening, A. (2010, September). Towards the automatic recognition of computational thinking for adaptive visual language learning. In *2010 IEEE symposium on visual languages and human centric computing* (pp. 59–66). New York, NY: IEEE.

Kolb, D. (1984). *Experiential learning: Experience as the source of learning and development*. Englewood Cliffs, NJ: Prentice Hall.

Kolb, D. (1999). The Kolb Learning Style Inventory, *Version 3*. Boston, MA: Hay Group.

Lee, Y. J. (2011). Scratch: Multimedia programming environment for young gifted Learners. *Gifted Child Today*, *34*(2), 26–31.

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

Lewis, C. M. (2011). Is pair programming more effective than other forms of collaboration for young students?. *Computer Science Education*, *21*(2), 105–134.

Lewis, A., & Smith, D. (1993). Defining higher order thinking. *Theory Into Practice*, *32*, 131–137.

Liao, Y. K., & Bright, G. (1991). Effects of computer-assisted instruction and computer programming on cognitive outcomes: A meta-analysis. *Journal of Educational Computing Research*, *7*(3), 251–268.

Metz, S. S. (2007). Attracting the engineering of 2020 today. In R. Burke, M. Mattis, & E. Elgar (Eds.), *Women and Minorities in Science, Technology, Engineering and Mathematics: Upping the Numbers* (pp. 184–209). Northampton, MA: Edward Elgar Publishing.

Mioduser, D., Levy, S., & Talis, V. (2009). Episodes to scripts to rules: Concrete abstractions in kindergarten children's explanations of a robot's behaviors. *International Journal of Technology and Design Education*, *19*(1), 15–36.

Moomaw, S. (2012). STEM begins in the early years. *School Science and Mathematics*, *112*(2), 57–58.

Next Generation Science Standards https://www.nextgenscience.org/ (accessed 15 November 2017)

National Academy of Sciences. (2007). *Beyond bias and barriers: Fulfilling the potential of women in academic science and engineering*. Washington, DC: The National Academies Press.

National Academy of Sciences. (2010). *Rising above the gathering storm, revisited: Rapidly approaching category 5*. Washington, DC: The National Academies Press.

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

Organisation for Economic Co-operation and Development. (2015). Schooling redesigned: Towards innovative learning systems Retrieved from http://www.oecd-ilibrary.org/education/schooling-redesigned_9789264245914

Oregon Department of Education. (2015). *School and district report cards*. Retrieved from http://www.ode.state.or.us/data/reportcard/reports.aspx

Papert, S. (1991). Situative constructionism. In I. Harel & S. Paper (Eds.), *Constructionism* (pp. 1–11). Norwood, NJ: Ablex.

Patton, M. Q. (2002). *Qualitative research and evaluation methods* (3rd ed.). Thousand Oaks, CA: Sage.

Reppening, A., Webb, D., & Ioannidou, A. (2010, March). Scalable game design and the development of a checklist for getting computational thinking into public schools. In: *Proceedings of the 41st SIGCSE technical symposium on computer science education (SIGCSE' 10)* (pp. 265–269). New York, NY: ACM Press.

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

Rushkoff, D. (2010). *Program to be programmed: Ten commands for a digital age*. New York, NY: O/R Books.

Scaffidi, C., & Chambers, C. (2012). Skill progression demonstrated by users in the Scratch animation environment. *International Journal of Human-Computer Interaction*, *28*(6), 383–398.

Sengupta, P., Kinnebrew, J. S., Basu, S., Biswas, G., & Clark, D. (2013). Integrating computational thinking with K-12 science education using agent-based Computation: A theoretical framework. *Education and Information Technologies*, *18*, 351–380.

Strauss, C. (1992). What makes Tony run? Schemas as motive reconsideration. In R. D'Andrade & C. Strauss (Eds.), *Human Motives and Cultural Models* (pp. 191–224). Cambridge, England: Cambridge University Press.

Strauss, C., & Quinn, N. (1997). *A cognitive theory of cultural meaning*. Cambridge, England: Cambridge University Press.

Weintrop, D., Beheshti, E., Horn, M. S., Orton, K., Joan, K., Trouille, L., & Wilensky, U. (2014, April). *Defining computational thinking for science, technology, engineering, and math*. Philadelphia, PA: Poster presented at the Annual Meeting of the American Research Association, PA.

Werner, L., Denner, J., Campe, S., & Kawamoto, D. C. (2012, February). The fairy performance assessment: Measuring computational thinking in middle school. In *Proceedings of the 43rd ACM technical symposium on computer science education* (pp. 215–220). New York, NY: ACM.

Wing, J. (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*, *60*(4), 55–62.

Author Biography

Yune Tran is an associate professor of Education at George Fox University. Her research interests include teacher efficacy and identity, ESL methods and practices, diverse learners, computational thinking in elementary contexts, and preservice teacher preparation.