Computers & Education 169 (2021) 104222

Contents lists available at ScienceDirect

Computers & Education

journal homepage: www.elsevier.com/locate/compedu

Learning to code and the acquisition of computational thinking by young children

E. Relkin *, L.E. de Ruiter, M.U. Bers

Eliot-Pearson Department of Child Study and Human Development, Tufts University, USA

ARTICLE INFO

Keywords:
Computational thinking
Coding
Early childhood education
Unplugged assessment
Curriculum

ABSTRACT

This longitudinal study examined changes in Computational Thinking (CT) skills in first and second grade students exposed to a developmentally appropriate coding curriculum. The "Coding as Another Language" (CAL) curriculum spans seven weeks and uses the KIBO robot to engage students in learning that integrates programming and literacy concepts. We compared children receiving CAL ($N = 667$) to a control group ($N = 181$) who participated in typical classroom activities without coding (No-CAL). *TechCheck,* a validated "unplugged" CT assessment suitable for young children regardless of their coding experience, was used to measure CT. Over the course of the study, children who received CAL-KIBO improved on *TechCheck* ($M_{change} = 0.94$, $p < .001$) whereas the No-CAL group did not change significantly ($M_{change} = 0.27$, $p = .07$). Accounting for demographic factors, baseline performance and classroom (teacher) effects, CAL exposure was a significant predictor of post-test CT scores ($p < .01$). Improvements in CT measured by *TechCheck* over seven weeks of the CAL-KIBO curriculum were consistent with approximately six months of development without coding instruction. Secondary analysis stratified by grade revealed decisive evidence that CAL exposure improved scores in first grade and anecdotal evidence that second grade scores improved. The CT domains that showed improvement in children who received CAL-KIBO included algorithms, modularity, and representation. Young children who learned to code improved in solving unplugged problems that were not explicitly taught in the coding curriculum. This provides evidence that a developmentally appropriate curriculum for teaching young children to code can accelerate their acquisition of CT skills.

1. Introduction

One of the most important goals of teaching computer science (CS) to young children is to foster the development of computational thinking (CT) skills that are applicable to many educational disciplines and areas of life (Barr & Stephenson, 2011; Chen et al., 2017; Cuny et al., 2010; Wing, 2006). Papert (1980) alluded to CT in his book *Mindstorms* in a discussion of the challenge of integrating Computer Science (CS) education with children's everyday experiences. Later, Wing popularized the term and defined it as a set of reasoning skills for formulating and solving problems using computers and other information technologies (Wing, 2006, 2011). She emphasized that CT is not only useful in CS but also other disciplines such as mathematics, science, design, economics, and linguistics (Wing, 2011). Since that time there has been increasing interest in CT, as documented in several recent reviews describing CT's definitions, methods of assessment and educational initiatives (Lye & Koh, 2014; Román-González et al., 2019; Tang et al., 2020;

* Corresponding author. Eliot-Pearson Department, 105 College Ave, Medford, MA, 02155, USA.
E-mail address: Emily.relkin@tufts.edu (E. Relkin).

https://doi.org/10.1016/j.compedu.2021.104222
Received 3 September 2020; Received in revised form 19 April 2021; Accepted 21 April 2021
Available online 29 April 2021

Zhang & Nouri, 2019).

There is an ongoing debate about whether CT is truly a singular concept (Barr et al., 2011; Grover & Pea, 2013; National Research Council, 2011). Zhang and Nouri (2019) identified three types of definitions of CT in the published literature: *generic definitions* that focus on universal problem-solving skills (e.g., Aho, 2012; Wing, 2011); *operational definitions* that provide a vocabulary and identify CT sub-domains (e.g., CSTA, 2011; Selby & Woollard, 2013) and *educational definitions* that provide concepts and competencies (e.g., Barr & Stephenson, 2011; Brennan & Resnick, 2012). Tang et al. (2020) later distinguished CT definitions that are programming-related (e.g., Grover, et al., 2015) from those of a general problem-solving nature (e.g., CSTA, 2011; Selby & Woollard, 2013). The existence of many different definitions is an indication that CT is still an evolving concept but one recognized to have considerable importance for CS education. For present purposes, we define CT to be a set of heuristic reasoning skills that can be categorized into discrete sub-domains applicable to problem-solving in computer science and other disciplines.

Coding (programming) has been described as "the instrumental skill of CT" and "the primary means of teaching CT in primary school" (Arfé et al., 2019; Román-González, 2017; Wing, 2006). Programming languages are specifically designed to communicate instructions and solve problems with computers, and children as young as 3–4 years of age are capable of learning to code (Bers, 2018; Clements & Gullo, 1984; Kazakoff & Bers, 2014; Strawhacker & Bers, 2019). However, in a 2014 review Lye and Koh (2014) found that the majority of past studies of coding and CT were carried out in higher educational settings, and only 25% involved kindergarten through 12th grade students. Lockwood and Mooney (2018) conducted a systematic review of CT in secondary schools (children ages 11–18) and concluded that educational programs promoting CT in middle and high schools are becoming more widespread. While there has been an increase in CT educational initiatives and professional development programs for younger students and their teachers (Fraillon et al., 2018; Tang et al., 2020), more work is needed in this area. In particular, there is still only a limited understanding of the effects of learning to code on young children's cognitive development and how to best promote the development of CT.

1.1. Teaching computational thinking to young children

Educational initiatives relating to CT in young children must take into account the progression of cognitive development. A typically developing young child does not possess fully mature literacy, numeracy, and abstract reasoning skills (Piaget, 1971). According to developmental theorists, first and second grade children are typically in the preoperational or concrete operations stage. At the preoperational stage from around two years to six years of age, children tend to engage in concrete, egocentric thinking and are just beginning to develop knowledge about physical symbols and representation. By the concrete operations stage from approximately six to twelve years, they are better able to organize their thoughts, use logical reasoning skills, and rely less directly on physical representations of ideas (Bruner et al., 1966; Feldman, 2004; McDevitt & Ormrod, 2002; Piaget, 1953).

A young child's stage of development can constrain the CS concepts and CT skills they can readily master (Chen et al., 2017; Goldstein & Flake, 2016). For example, early elementary school children may have difficulty grasping "if-then" conditionals (Barrouillet & Lecas, 1999; Janveau-Brennan & Markovits, 1999; Muller et al., 2001). Likewise, they may have a hard time understanding abstract representations such as variables. They may engage in magical thinking or personification rather than recognize the mechanical basis for the actions of machines (Flavell et al., 1993; Mioduser et al., 2009). These and other developmental considerations must be taken into account when designing educational programs to teach CT to young children.

In an effort to provide a developmentally appropriate framework for teaching coding and other CS concepts to children between the ages of 4–9 years, Bers (2018) described the seven powerful ideas of CS. This framework is based on experience with a variety of coding initiatives for children, such as Google for Education, 2010; Scratch (Brennan & Resnick, 2012); the KIBO robotics kit (Sullivan & Bers, 2015) and ScratchJr (Portelance et al., 2015). The seven powerful ideas identify child-friendly concepts within the domains of hardware/software, algorithms, modularity, control structures, representation, debugging, and design process (see Table 1).

The powerful ideas provided the foundation for the CS curriculum used in the current study called "Coding as Another Language" (CAL). This curriculum is designed to teach coding and CT to young children while simultaneously promoting literacy skills (Bers, 2018; Hassenfeld et al., 2020). Programming in elementary education has typically been associated with Science, Technology, Engineering and Mathematics (STEM) curricula (Bers, 2019; Clements et al., 2001; Guzdial & Morrison, 2016). However, there are

Table 1
The seven powerful ideas, associated concepts, and examples from the CAL-KIBO curriculum.

Powerful Idea	Associated Concepts	Example from CAL-KIBO Curriculum
Algorithms	Sequencing/order, logical organization	Child learns to program KIBO in a specific sequence to dance the "Hokey Pokey"
Modularity	Breaking up larger task into smaller parts, instructions	Students break up the "If You're Wild and You Know It" song into smaller components that KIBO can be programmed to perform
Control Structures	Recognizing patterns and repetition, cause and effect	Children learn to trigger sound sensors using "wait for clap" command
Representation	symbolic representation, models	Child learns that each programming block translates into a unique KIBO action.
Hardware/ Software	Smart objects are not magical, objects are human engineered	Children play a game about what is and isn't a robot and learn that you must give the KIBO robot a program in order for it to perform
Design Process	Problem solving, perseverance, editing/ revision	Children are tasked with creating a final "Wild Rumpus" KIBO project in which they plan, code, test and revise with peer sharing and feedback
Debugging	Identifying problems, problem solving, perseverance	Children identify problems in either hardware or software of KIBO and brainstorm solutions to fix it

creative and self-expressive aspects of programming that align more closely with literacy and other aspects of the humanities (Bers, 2018, 2020; Resnick & Siegel, 2015). The CAL curriculum draws on principles of literacy education to create lessons that blend elements of learning to read and write with CS and coding concepts (Bers, 2019; Hassenfeld et al., 2020). This pedagogical approach emphasizes creative programming and provides children with opportunities for self-expression analogous to those experienced when using a symbolic written language (Bers, 2018, 2019).

One of the greatest challenges to integrating CT into early elementary school education has been a lack of validated, developmentally appropriate assessments to measure young children's CT skills in classroom and online settings (Lockwood & Mooney, 2018; Lee et al., 2011; Román-González et al., 2019). In the following section, we review the development of CT assessments for young children and describe the recent advent of "unplugged" CT assessments such as *TechCheck*, the instrument employed in this study.

1.2. Assessing computational thinking in young children

CT assessment instruments for young children must use developmentally appropriate language and tasks to assure that factors such as literacy and fine motor skills are not limiting (Chen et al., 2017; Sattler, 2014). Cultural biases should be avoided, and the activities and artifacts employed must be familiar and non-threatening to young children (McMillan, 2013; Mullis & Martin, 2019; Tang et al., 2020). The duration of the assessment should be relatively brief in light of the shorter attention span of young children (Moyer & Gilmer, 1953). The range of difficulties covered by the assessment should allow for children with little or no CT training to be assessed with equal ease and precision to students with extensive CT talent (Relkin et al., 2020). It has been suggested that CT assessments should incorporate measures that evaluate reasoning processes, not just the end product of a program or a problem solved (Brennan & Resnick, 2012; Fields et al., 2019; Román-González et al., 2019). However, this is arguably an aspirational goal that has yet to be achieved in a brief CT assessment that can be administered to large numbers of young students simultaneously in a classroom setting.

Román-González et al. (2019) reviewed CT assessment tools for kindergarten through 12th grade and found most were designed for students in middle school, high school and/or adults (Chen et al., 2017; Fraillon et al., 2018; Román-González et al., 2018; Werner et al., 2012). Some CT assessments require hours and/or multiple sessions to complete, making them impractical for routine use in educational settings (Basu et al., 2016; Chen et al., 2017; Werner et al., 2014). CT assessments that employ programming challenges that require some prior knowledge of coding may conflate programming abilities with CT skills (Yadav et al., 2017). Such instruments cannot readily be used to assess baseline CT abilities in coding-naive students. To the extent that it is desirable to be able to measure CT skills in children regardless of whether they have past knowledge or experience with computer programming, coding exercises alone may not be the best way to assess CT (Grover et al., 2014).

Recently, our research group and others have explored the use of coding-free instruments to assess CT skills in children. These newer instruments leverage the fact that CT skills can be exercised without programming through the use of unplugged activities (Bell & Vahrenhold, 2018; Zapata-Cáceres et al., 2020). Unplugged activities consist of puzzles, games and other exercises that draw upon CS concepts without requiring explicit knowledge of coding or computers. Unplugged activities have been used to teach CS concepts for over two decades (e.g., CSUnplugged.com; code.org) and can also be used for assessment purposes.

One of the first unplugged CT assessments was designed for post-elementary school students by Román-González et al. (2018) who created a 45-min unplugged assessment called the Computational Thinking Test (CTt). This instrument was used to measure CT abilities in over 300 middle school students (ages 12–14) before and after they took part in an informatics course that included elements of the code.org curriculum (Román-González et al., 2018). After the coding intervention, CTt assessment scores improved and correlated positively ($p < .01$) with language ($r = 0.42$), grade point average ($r = 0.47$), mathematics ($r = 0.36$) and informatics ($r = 0.43$). In its original form, the CTt is not suitable for use in younger, elementary school-age children.

Arfé et al. (2019) used four traditional neuropsychological tests to measure executive functioning in first and second graders who received components of the code.org curriculum. The tests did not involve coding or computer technology and as such could be considered "unplugged." Among $n = 42$ first graders who received 8 h of coding instruction, the measures of response inhibition and planning improved more than in a control group ($n = 34$) that received non-coding STEM activities (Arfé et al., 2019). The authors also followed $n = 17$ second grade students longitudinally and found that changes in planning and response inhibition after one month of

Table 2
Comparison of Two "Unplugged" CT assessments for young children: The BCTt and *TechCheck* based on Zapata-Cáceres et al., 2020 and Relkin et al., 2020.

	BCTt	*TechCheck*
Average Admin Time	40 min	13 min
Format	Pen and paper	Pen and paper, Online
Validation Sample	299 students	768 students
Validated Age Range	5-12 (1st- 6th grade)	5-9 (1st - 2nd grade)
Age Sensitivity	Significant difference between 2nd grade vs. 4th and 6th graders in initial validation study. No significant difference between 1st and 2nd graders reported. No difference between 4th and 6th graders	Significant difference between 1st and 2nd graders. No data on older or younger children in initial validation study.
CT Concepts	Sequences, Loops (Simple, Nested), Conditionals (If-Then, If-Then-Else, While)	Algorithms, Modularity, Debugging, Hardware/Software, Control Structures, Representation

the coding intervention were comparable to those that occurred over seven months of normal development. This study provides evidence from a randomized, control trial that learning to code can accelerate the development of executive functions critical to CT in young children. However, the number of participants was relatively small and the assessment measures employed focused on a specific subset of the various skills involved in CT. In light of this, further studies are needed to evaluate the impact of learning to code on young children's CT skills.

Cross-sectional validation studies were recently completed on two unplugged CT assessments designed specifically for young children. The *CTt for Beginners* (BCTt) (Zapata-Cáceres et al., 2020) and *TechCheck* (Relkin et al., 2020) both use unplugged challenges to probe CT domains and can be administered to children who lack prior coding experience. These instruments differ in the types of unplugged challenges they include, the CT domains assessed, the targeted age ranges and the time required to complete the assessments (see Table 2).

Although both the BCTt and *TechCheck* assessment instruments have unique merits, *TechCheck* was chosen for the present study for several reasons. *TechCheck*'s CT constructs are based on Bers' seven powerful ideas, the same conceptual foundation as the CAL coding curriculum used in this study. On average, *TechCheck* takes approximately 13 min to administer while the BCTt requires approximately 40 min. Some of the concepts probed by the BCTt such as conditionals may be problematic for younger children on developmental grounds (Barrouillet & Lecas, 1999; Janveau-Brennan & Markovits, 1999; Muller et al., 2001). Mean scores on *TechCheck* were significantly different in first and second graders whereas no significant difference between these grades was reported for the BCTt (Zapata-Cáceres et al., 2020). In addition, *TechCheck* can be administered to large groups of children simultaneously using an online platform, which is useful in the context of the present study involving hundreds of students.

By obtaining a better understanding of how learning to code impacts the acquisition of CT in young children, it may be possible to improve teaching methods designed to promote the development of these reasoning skills (Nouri et al., 2020). The advent of the CAL curriculum and validated unplugged CT assessments for elementary school children provides a new opportunity to explore the interaction of coding education and CT. We set out to answer the following research question: How does a coding intervention impact young children's CT skills as measured by an unplugged CT assessment?

2. Method

The present study has a quasi-experimental longitudinal design. The intervention is a version of the CAL curriculum called "CAL-KIBO" that uses the KIBO robot to teach children programming and literacy concepts. It examines CT skills in children between ages 5 and 9 (first and second grade) before and after they participate in the CAL-KIBO curriculum. Grade-matched students who engage in their usual classroom activities without learning to code provide a comparison group for identifying incidental and/or maturation-related changes in CT skills. The *TechCheck* unplugged assessment is administered before and after the intervention to evaluate changes in CT. In the following section, we describe the specifics of the methods we employ.

2.1. The intervention: the CAL-KIBO curriculum

The CAL-KIBO curriculum is implemented using the KIBO robotics platform, a screen-free programmable robot that is developmentally appropriate for young children. Young children often learn to code using simple sequencing and graphical or tangible coding interfaces (Bers, 2020; Guzdial & Morrison, 2016; Jenkins, 2002; Resnick & Silverman, 2005; Strawhacker et al., 2017; Sullivan et al., 2015). KIBO is programmed with tangible wooden blocks that a child sequences and then scans using a barcode scanner embedded in the robot. Each block represents an action that the robot performs. The combination of KIBO's blocks, sensors, modules, and art platforms gives children a unique opportunity to not only explore programming concepts but also to use their creativity to create personally meaningful projects (see Fig. 1).

Fig. 1. The KIBO robot, programming blocks, parameter stickers, modules/sensors, and attachable art platforms.

KIBO has been shown to engage young children as young as four years old in expressive and creative coding (Elkin et al., 2016; Sullivan et al., 2015, 2017). The CAL-KIBO curriculum incorporates lessons and exercises that teach algorithms, modularity, hardware/software, control structures, debugging, representation, and design process.

The CAL-KIBO curriculum teaches coding as a symbolic system of representation for expressive purposes and not only problem-solving. CAL-KIBO includes time spent working with coding, game-play as well as an emphasis on activities involving social interactions, creativity and movement. Individual and group activities in this curriculum include warm-up games to playfully introduce or reinforce concepts, design challenges to solidify skills, free explorations to allow students to tinker and expand their skills, expressive explorations to promote creativity, writing activities and technology circles to share and reflect on activities. The curriculum was first created and tested with second graders and was then modified for first graders. Feedback from teachers, administrators and students was taken into account when designing this curriculum. The CAL-KIBO curriculum is aligned with the Common Core English Language Arts (ELA)/Literacy Framework, as well as Virginia CS Standards of Learning and other nationally recognized CS frameworks (ISTE Standards for Students, 2017; K-12 Computer Science Framework Steering Committee, 2016; Massachusetts Department of Elementary and Secondary Education, 2016; National Governors Association Center for Best Practices & Council of Chief State School Officers, 2010; Virginia Department of International Society for Technology in Education, 2017).

The second grade CAL-KIBO curriculum consisted of 12 1-h lessons. Lessons were designed to be carried out in 1–2 h of instruction each week over 6–7 consecutive weeks. Each lesson consisted of structured KIBO challenges, opportunities for free exploration and writing activities. The advanced programming concepts in this curriculum included repeat loops, the use of light and distance sensors, and conditionals. The final lesson involved a multi-day project based on the popular children's book *Where the Wild Things Are* by Maurice Sendak, which was referenced at several points throughout the curriculum. This book was chosen because it fosters discussion and creative thinking and allows teachers to integrate literacy and computer science concepts into their lessons.

The first grade CAL-KIBO curriculum followed the same implementation timeline and used the same story *Where the Wild Things Are* and covered much of the same KIBO concepts but did not cover conditional statements. Additionally, 3 h of additional lesson time were added to the original 12-h curriculum based on teacher feedback. The "Wild Rumpus" compositional activity was omitted so that students could focus more on programming and student-centered discussions with KIBO. First grade teacher and classroom support materials were enhanced to better assist teachers in implementing the curriculum.

An example of a CAL lesson involves one of the main scenes in *Where the Wild Things Are* consisting of six pages of illustrations showing the main character Max participating in a "Wild Rumpus Party." Students were asked to write a creative composition about what would happen at their own Wild Rumpus Party. The class then discussed their compositions as a group and collaborated with one another to decide whether or not what they had written about could be rendered as a program for KIBO to perform. Children then programmed the KIBO to perform their Wild Rumpus party activities (see Fig. 2A). For example, one child wrote that her KIBO would sing karaoke and dance. The child used stickers corresponding to those on the KIBO blocks to plan her program and subsequently programmed KIBO using actual programming blocks and a recording of her own singing made using the KIBO sound recorder module.

2.2. Participants

Participants in this study were first and second graders from an urban school district in Norfolk, Virginia. Students were from military and non-military families with a mixture of different racial/ethnic and socio-economic backgrounds (Table 3). Ten schools were invited to participate in this study. Eight of the ten schools received a grant from the U.S. Department of Defense and were chosen to receive the CAL-KIBO curriculum. Two additional schools were included for comparison purposes. Students from the No-CAL control schools followed a standard curriculum without exposure to CAL or coding. No-CAL students underwent assessments at comparable time intervals to the CAL schools. The No-CAL schools had similar overall demographics to the schools that received CAL (Table 3).

Among the eight schools invited to implement the CAL curriculum, two schools contributing first graders did not participate due to administrative and/or staffing issues. Among the No-CAL schools, one first grade class inadvertently received coding instruction during

Fig. 2. **A**: Students' CAL-KIBO final projects. **B:** KIBOs decorated by teachers at the CAL-KIBO training, *Note.* Photograph 2A courtesy of Angela de Mik, Norfolk Public Schools.

Table 3
Demographics of the study population.

	All CAL	All No-CAL	Grade 1 CAL	Grade 1 No-CAL	Grade 2 CAL	Grade 2 No-CAL
Number of students	667	181	271	71	396	110
Mean Age (Years)	7.41	7.38	6.23	6.28	7.56	7.61
Age Range (Years)	5–9	6–9	5–8	6–7	7–9	7–9
Gender						
Male (%)	47.20	42.54	48.34	43.66	46.46	41.81
Female (%)	51.87	56.35	50.92	56.34	52.53	56.36
Not specified (%)	0.93	1.01	0.74	0	1.01	1.82
Race						
Black/African American (%)	41.25	53.59	35.42	57.75	45.21	50.91
Hispanic (%)	10.19	14.26	10.33	16.90	10.10	12.73
Mixed (%)	8.54	5.52	9.59	5.63	7.83	5.45
White (%)	36.58	24.97	40.59	18.32	33.83	29.09
Asian/Pacific Islander (%)	2.99	1.66	3.32	1.40	2.78	1.82
Native American (%)	0.45	0	0.74	0	0.25	0

the study window in violation of the study protocol and was excluded from the analysis.

2.3. Inclusion criteria

Inclusion criteria for this study were: 1. parental opt-out consent and child assent; 2. adequate English language skills to participate in the curricular activities and study assessment. Inclusion criterion for the main analysis was the completion of baseline and end point *TechCheck*. Inclusion criterion for the baseline analysis was the completion of the pre-CAL *TechCheck* assessment.

2.4. Professional development

Educators attended a full day, CAL-KIBO training led by multiple researchers where they participated in hands-on play with the KIBO robot and were introduced to the CAL approach and curriculum. At the training, teachers participated in activities from the curriculum such as creating their own *Where the Wild Things Are* KIBO final projects (see Fig. 1B). Opportunities were provided to practice and plan for classroom implementation. Teachers were given hard copies of the curriculum and children's books to aid their instruction as well as online resources such as videos of others teaching the curriculum, links to the curriculum lessons, and lesson slides. Ongoing professional development and support was given to educators through phone calls with researchers and in-person assistance from administration staff, researchers, and instructional technology resource teachers.

Instructional Technology Resource Teachers (ITRTs) from the school district attended both the CAL-KIBO training and separate in-person assessment workshops. At the assessment workshops, ITRTs were taught to administer *TechCheck* including what to do in various scenarios (e.g., a child needing to leave the room, or asking them if they got the correct answer). ITRTs were given time to practice administration. A log was created to keep track of which classes received assessments and when. Additionally, throughout the study ITRTs and researchers engaged in multiple phone conferences to provide feedback on assessment administration.

2.5. Computational thinking assessment

The *TechCheck* assessment used in this study consists of fifteen multiple-choice questions. *TechCheck* is considered an "unplugged" assessment because its challenges probe CT but do not require the use of technology or knowledge of computer programming to be completed. In the present study, *TechCheck* was administered using computers and tablets rather than pencil and paper. However, it is the content rather than the mode of administration that leads to the characterization of *TechCheck* as an unplugged assessment. The child responds to prompts on *TechCheck* by clicking on one of four options. Each correct response is awarded one point, with a maximum total score of 15 points. Two practice questions are included in the beginning of the assessment to familiarize students with the format but are not included in the scoring. All questions must be answered to complete the assessment. The *TechCheck* assessment typically takes an average of 13 min for children to complete. *TechCheck* was previously validated with a sample ($N = 768$) of 5-9-year-old children in first and second grade (Relkin et al., 2020). The assessment showed good discrimination of children between different skill levels and an adequate difficulty level for first grade. The difficulty level for second graders was low and a ceiling effect was evident for the highest performers. Children's scores on *TechCheck* correlated moderately and positively ($r = 0.53$) with a CT measure (TACTIC-KIBO) that requires knowledge of coding with the KIBO robot (Relkin et al., 2020).

TechCheck probes six of the seven powerful ideas from computer science described by Bers (2018) as developmentally appropriate for children ages 4–9. This includes algorithms, modularity, control structures, representation, hardware/software, and debugging. Design process, the seventh powerful idea, was not included in *TechCheck* because it is an inherently open-ended process that cannot be readily measured in a multiple-choice format assessment (Relkin et al., 2020). A variety of different tasks are used to probe the six CT domains: sequencing challenges, shortest path puzzles, missing symbol series, object decomposition, obstacle mazes, symbol shape puzzles, identifying technological concepts, and symmetry problems (see Appendix).

2.6. Procedure

After attending in-person professional development, the teachers were given two weeks to prepare their classroom schedules. One week prior to initiating the curriculum, the *TechCheck* assessment was administered by one of eight ITRT proctors (one per school). The endpoint *TechCheck* was given after the full curriculum had been taught.

ITRTs were trained to administer the *TechCheck* assessments consistently. Entire classrooms were tested together on individual tablets. Before children arrived, ITRTs prepared enough devices for the classroom and opened the *TechCheck* assessment application saved to the desktop. ITRTs first established rapport with children, then asked children for their assent to participate. Since the *TechCheck* assessment is designed for use in children who may be pre-literate or marginally literate, administrators were instructed to project a copy of the assessment onto a board and read each question out loud to the students twice. There were two practice questions that the classroom did as a group to ensure that children knew how to use the application and select answers using the interface. Students were then told to work individually and were given up to 1 min to answer each question. Each question required a response and children were instructed to guess if they did not know the answer.

2.7. Data analysis

Statistical analyses were conducted in R (Version 3.6.1, R Core Team, 2019) using R Studio version 1.2 (R Core Team, 2019). To assess longitudinal changes over time, we analyzed the data with a General Linear Mixed Model (GLMM) using the "lme4" package (Bates et al., 2015) and the "lmerTest" package (Kuznetsova et al., 2017). Pre-analysis data screening showed adequate normality of the variables used in the models. The GLMM fixed effects were: CAL vs No-CAL status, age, self-reported gender, grade, and Baseline *TechCheck* score. The random effect was classroom/teacher. *P*-values were obtained by likelihood ratio tests for the full model with CAL and the model without CAL and by using the "summary" function on R studio. A post-hoc analysis of residuals showed adequate normality on histogram and P–P plots. VIF and Tolerance assumptions of multicollinearity were met. Although the majority of the data appeared to obey the assumptions of homoscedasticity, there was some sparseness of data at the lower end of the range of *TechCheck*

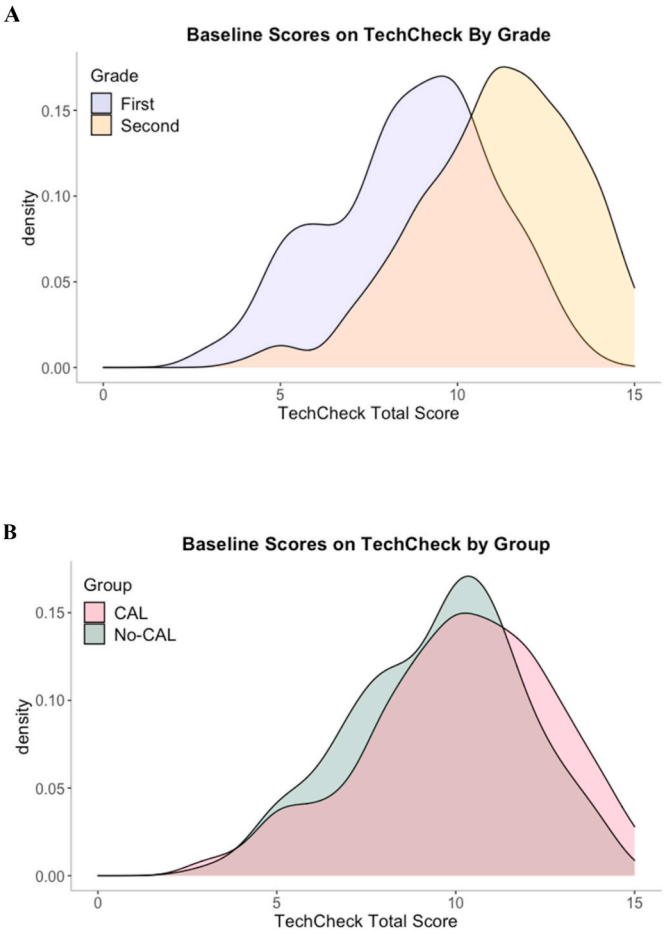

Fig. 3. The Distribution of Baseline *TechCheck* Scores by grade(3A) and by group (3B).

scores, so the assumption of homoscedasticity may not have been met. No influential outliers were identified.

We calculated the expected rate of change of *TechCheck* scores over time in the absence of a coding intervention by subtracting the mean baseline *TechCheck* score in first graders from the mean score in second graders and dividing that by the difference in mean ages between the two grades.

To further assess CAL-KIBO's contributions to the study outcomes, we conducted Bayesian Linear Mixed Modeling (BLMM) with the "BayesFactor" package (Morey & Rouder, 2018). BLMM was used in this study to supplement classical hypothesis testing since one cannot infer from this type of analysis whether the null hypothesis is true if, for example, results are non-significant (see Dienes, 2014). Bayesian analyses can provide information about the relative strength of the statistical evidence for both the null and alternative hypotheses. A Bayes factor is the likelihood ratio of one hypothesis divided by that of another hypothesis.

A post-hoc item analysis was carried out examining change in the percentage of correct responses in the six *TechCheck* CT domains. First, the percentages of correct responses on each of the 15 *TechCheck* items were calculated and then averaged within each CT domain. Baseline percentages were then subtracted from endpoint percentages to determine change over the course of the 7-week intervention for the students who received the CAL-KIBO curriculum (CAL) and those who did not (No-CAL). For comparison purposes, predicted change over seven weeks of typical development was calculated by subtracting the percentage of correct responses for all first graders at baseline from those of all second graders at baseline, and multiplying the resulting percentages by 7/67.8 to adjust for the 1.3-year average difference in age between the two grades.

3. Results

3.1. Baseline score distributions

The distributions of *TechCheck* scores at baseline for first and second grades are shown in Fig. 3a. Scores were approximately normally distributed. A rightward skew is visible in the second grade distribution, consistent with the ceiling effect on *TechCheck* previously observed in second graders (Relkin et al., 2020). A Welch Two Sample *t*-test showed that there was a significant difference in baseline *TechCheck* scores between first ($M = 8.45$ points, $SD = 2.33$) and second ($M = 10.99$ points, $SD = 2.20$) grades; $t\,(703.81) = 15.92$, $p < .001$. We used the mean difference in baseline scores (2.54 points) divided by the mean difference in the ages of first and second graders (1.30 years) to calculate the approximate expected change in *TechCheck* scores between first and second grade (1.95 points/year). Since most of the study participants (>75%) indicated they had little or no past coding experience, this rate of change can be taken to approximate maturation-related changes in CT skills over time.

Fig. 3b shows the distribution of scores for students in the CAL and No-CAL groups. A Welch Two Sample *t*-test showed that baseline performance was higher in the schools that received CAL ($M = 10.09$ points, $SD = 2.61$) compared to the No-CAL schools, ($M = 9.50$ points, $SD = 2.38$; $t\,(307.73) = 2.93$, $p < .01$). This difference in baseline scores occurred by chance rather than by design.

3.2. Primary outcome

Students who received the CAL-KIBO curriculum (CAL group) improved on *TechCheck* ($M_{change} = 0.94$, $SD = 2.28$). Paired sample t-tests showed that the CAL group's change from baseline ($M = 10.09$, $SD = 2.61$) to endpoint ($M = 11.03$, $SD = 2.61$) was significant; $t\,(666) = 10.55$, $p < .001$. Students in the No-CAL control group who engaged in typical classroom studies without coding instruction did

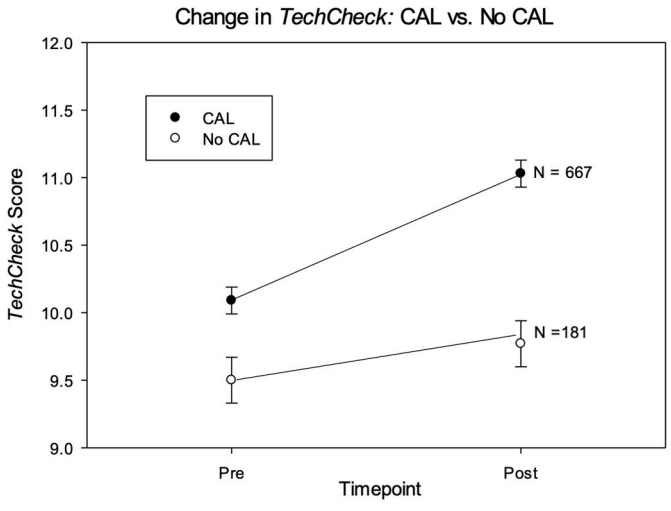

Fig. 4. Unadjusted mean change in *TechCheck* scores (+/− S.E.) from baseline (pre) to endpoint (post) in students receiving the CAL-KIBO coding curriculum ● versus control (No-CAL) students ○.

not significantly improve ($M_{change} = 0.27$ points, $SD = 2.04$) from baseline ($M = 9.50$, $SD = 2.38$) and the study endpoint ($M = 9.77$, $SD = 2.55$) assessments; $t(180) = 1.81$, $p = .07$) (see Fig. 4).

The observed change of 0.94 points in the CAL group equates to the increase in *TechCheck* scores that would be expected to occur over approximately 6 months without instruction based on the observed increase of 1.95 points per year in baseline scores. The change of 0.27 points in the No-CAL control group is consistent with the expected change over seven weeks, which is the actual interval over which the testing was performed.

Results were stratified by grade and paired sample t-tests were conducted. In first grade the CAL group significantly improved ($M_{change} = 1.32$ points, $SD = 2.31$) from baseline ($M = 8.63$, $SD = 2.35$) to endpoint ($M = 9.95$ $SD = 2.34$) of the study; $t(270) = 9.21$, $p < .001$. Likewise, the second grade CAL group significantly improved ($M_{change} = 0.68$ points, $SD = 2.23$) from baseline ($M = 11.15$ $SD = 2.20$) to the study endpoint ($M = 11.83$ $SD = 2.46$); $t(395) = 6.11$, $p < .001$. No significant improvements were found for the No-CAL control group in first grade ($M_{change} = 0.01$, $SD = 2.10$) from baseline ($M = 8.03$, $SD = 2.05$) to endpoint ($M = 8.10$, $SD = 2.27$) assessments; $t(358.31) = 1.07$, $p = .95$). A borderline significant improvement was found in the second grade No-CAL control group ($M_{change} = 0.44$, $SD = 2.00$) from baseline ($M = 10.43$ $SD = 2.10$) to endpoint ($M = 10.87$, $SD = 2.07$); $t(109) = 2.34$, $p = .05$), (see Table 4).

3.3. GLMM results

To take into account baseline differences and to evaluate the contribution of other effects such as age, gender, ethnicity and classroom differences, we used Generalized Linear Mixed Model (GLMM) analysis. For the two grades combined, the effects found to be significant on the GLMM included the intercept ($p < .001$), CAL ($p < .01$), grade ($p < .01$) and baseline *TechCheck* score ($p < .001$) (See Table 5 and Fig. 5). Gender and age did not significantly contribute to the model. This indicates that exposure to the CAL-KIBO curriculum was a significant predictor of endpoint *TechCheck* outcome, even when taking into account differences in baseline *TechCheck* performance and other effects. The Bayes factor for this model compared to a model without CAL was >100. This is classified as "decisive evidence" against the null hypothesis (Wetzels et al., 2011) and strongly supports CAL being a significant factor in the overall *TechCheck* outcomes. Interaction terms did not improve the model's fit to the data and were therefore not included.

When data from the first grade students only were modeled, CAL ($p < .01$) and baseline score ($p < .001$) were significant effects. Gender, age, and the intercept were not significant. The Bayes factor for the model including CAL compared to one without CAL was >100, a level considered decisive evidence that exposure to CAL is a predictor of post-test CT scores. When data from second grade students were examined, gender was significant at the $p < .05$ level, as were the baseline score ($p < .001$) and the intercept ($p < .001$). Notably, CAL did not reach significance in this model. The Bayes factor, in this case was 1.88, which is considered "anecdotal evidence" (Wetzels et al., 2011). Thus, in contrast to the results in first graders, we cannot say with certainty that exposure to CAL is a predictor of *TechCheck* score in second graders.

3.4. TechCheck item analysis

We sought to determine if changes in *TechCheck* scores after exposure to the CAL-KIBO curriculum were related to improved performance in specific CT domains. To do so, we carried out a post-hoc analysis of change in percentage of correct responses averaged over the questions within each CT domain. Fig. 6 shows the change in percentage of students whose scores improved from the study baseline to endpoint on each of the six domains measured by *TechCheck*.

All *TechCheck* CT domains showed change with typical development as calculated from differences between first and second grades at baseline. Hardware/software and debugging showed slightly less change with typical development than the other domains. There was likely a ceiling effect for these two domains since an average of 90% of students responded to hardware/software probes correctly and 87% responded correctly to debugging probes at baseline.

Across all six *TechCheck* domains, the average percentage increase in students responding correctly after the coding intervention was 6% for the CAL group and 2% for No-CAL controls. The largest percentage of student improvement in the CAL group was associated with the CT domains of Modularity, Algorithms and Representation, respectively (see Fig. 6).

Table 4
TechCheck results for CAL and No-CAL Control Groups.

Group	*N*	*TechCheck* Baseline ($M \pm$ SD)	*TechCheck* End Point ($M \pm$ SD)	Mean Points Changed	Median Points Changed	Paired *t*-test *p* Value	Bayes Factor (Bayesian *t*-test)
All CAL	667	10.09± 2.61	11.03± 2.61	0.94	1	$p < .001$	>1000
All No-CAL	181	9.50±2.38	9.77±2.55	0.27	0	$p = .07$	0.42
CAL Grade 1	271	8.63±2.35	9.95±2.34	1.32	1	$p < .001$	>1000
No-CAL Grade 1	71	8.06± 2.05	8.07± 2.28	0.01	0	$p = 0.95$	0.13
CAL Grade 2	396	11.15±2.20	11.83±2.46	0.68	1	$p < 0.001$	<1000
No CAL Grade 2	110	10.43±2.10	10.87±2.07	0.44	.5	$p < 0.05$	1.42

Table 5
GLMM results modelling exposure to CAL and outcome of *TechCheck*.

	Estimate	CI (95%)	Standard Error	*T*-value	*DF*	*P* value
Intercept	4.50	5.92	0.87	5.20	774.08	$p < .001$
CAL	0.87	1.33	0.29	3.05	56.13	$p < .01$
Age	−0.06	0.14	0.12	−0.46	817.23	$p = .05$
Grade	0.84	1.32	0.30	2.86	142.82	$p <. 01$
Gender	−0.19	0.03	0.13	−1.40	808.23	$p = .36$
Baseline score	.57	0.61	.03	18.51	831.59	$p < .001$

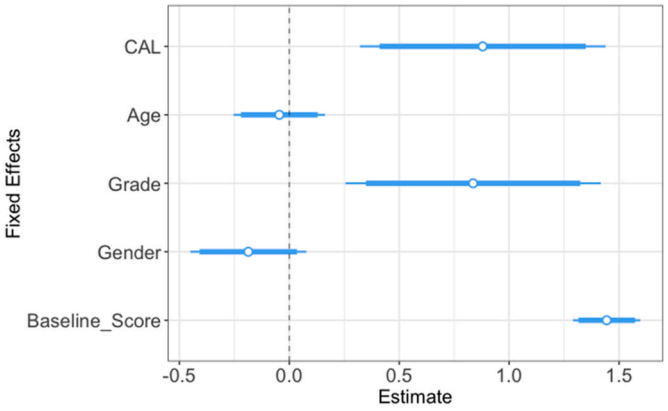

Fig. 5. Magnitude of effect and 95% C.I for fixed effects in GLMM

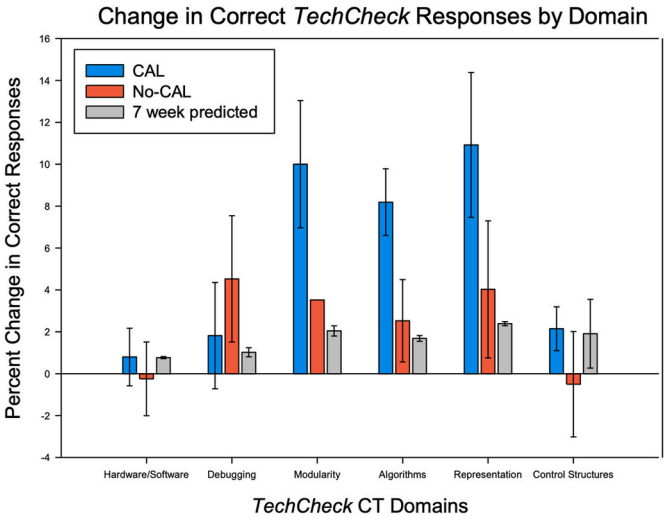

Fig. 6. Changes in percent of students making correct responses in the six CT domains measured by *TechCheck*. For the CAL and No-CAL groups, colored bars represent the percent difference in correct *TechCheck* responses between the baseline and end of study assessments. The "7 week predicted" results are calculated from the percent difference in correct responses between first and second graders at their respective baseline assessments, multiplied by 7/67.8. Error bars indicate standard errors of the means.

4. Discussion

This study provides empirical evidence, from a large-scale, quasi-experimental study, that teaching first and second grade students (ages 5–9) to code through the CAL-KIBO curriculum can accelerate the acquisition of CT skills. The current study is the first to use *TechCheck* to document longitudinal changes in CT in association with an educational intervention. The CAL-KIBO curriculum teaches coding without explicitly providing children with unplugged CT challenges of the kind encountered in *TechCheck*. Consequently, the observed increase in *TechCheck* scores following the CAL-KIBO coding curriculum can be taken to reflect improvements in CT skills rather than practice effects or other assessment artifacts. This conclusion is further bolstered by the observation that the No-CAL group

did not demonstrate comparable improvement on *TechCheck* over the same time interval.

Most past studies in young children that explore the effects of learning to code on CT skills used coding exercises, interviews or measures such as neuropsychological tests as the means of assessment (Grover et al., 2014; Yadav et al., 2017). By utilizing an easily administered unplugged CT assessment, we were able to conduct the first longitudinal large-scale study in early elementary school children with a sizable non-coding control group. Our approach allowed assessment of baseline CT skills in these children regardless of their past coding experience and avoided conflating coding abilities with CT skills. The study also demonstrates the potential utility of *TechCheck* for assessing young children's CT in routine education settings.

The mean change in *TechCheck* scores in students exposed to the CAL-KIBO coding curriculum was slightly less than one point out of a maximum score of 15 points. This magnitude of improvement after seven weeks of coding instruction is consistent with the estimated change in baseline *TechCheck* scores in the absence of coding instruction over approximately six months. This change is comparable in magnitude to the improvements in executive functions reported by Arfé et al. (2019) in a group of $n = 17$ second graders, in whom exposure to the code.org curriculum for one month resulted in improvements in executive functions equal to 7 months in age-matched students who received non-coding STEM instruction (Arfé et al., 2019).

Our data also indicate that young children's performance on a CT assessment can improve in the course of typical development. CT may improve as a consequence of brain maturation, practice effects, as a result of learning in other disciplines and through various life experiences. The present study does not allow us to say whether CT acquired in the context of learning to code is the same or different in nature from that gained through typical development or non-coding experiences. However, by accelerating the acquisition of CT, coding interventions in early childhood may exert long-term benefits analogous to the improved academic outcomes associated with early acquisition of literacy skills (Heckman & Masterov, 2007; Stanovich, 1986, 2000).

In this study, first graders in the CAL group improved more on *TechCheck* than the CAL group second graders. It is possible that the higher baseline *TechCheck* scores in second graders reduced the range of possible improvement compared to first graders. Previous psychometric analysis of the *TechCheck* assessment revealed a less-than-optimal difficulty level for higher performing second graders (Relkin et al., 2020) which may have led to a ceiling effect in the present study. Another possible explanation for the observed differences between grades would be if the CAL-KIBO curriculum was more effective in first graders. This possibility can neither be confirmed nor ruled out based on the currently available data.

There were significant differences at baseline in the *TechCheck* scores of the CAL-KIBO versus the No-CAL control groups. We carried out GLMM modeling to take those differences into account and evaluate the effects of other demographic and environmental variables on the study's outcomes. The results of the GLMM analyses indicate that exposure to CAL-KIBO was a highly significant predictor of *TechCheck* outcome even when taking the other variables into account. GLMM analysis did not confirm a significant effect for CAL in second graders which we believe is due at least in part to a ceiling effect in that grade discussed above.

The inclusion of a control group that did not receive coding instruction is important for several reasons. This is the first study in which *TechCheck* was administered serially to large numbers of students, and it was therefore important to control for the possibility of a learning effect from repeated exposure to the assessment. The results from the No-CAL control group suggest that any learning effects from repeated testing did not profoundly affect the study's outcome. The inclusion of a No-CAL control group also allowed for observation of changes in *TechCheck* performance related to the maturation of students over the time interval of the study.

Our findings suggest that some students who received coding instruction were able to transfer the knowledge they gained from coding into CT skills useful for solving unplugged problems. Although *TechCheck* is not designed to quantitatively assess CT skills in specific CT subdomains, post hoc analysis suggests that the domains of CT that improved most after the CAL-KIBO curriculum were algorithms, modularity, and representation. These are similar in nature to the domains identified in surveys of educators as being enhanced in young children who learn to code (Nouri et al., 2020).

It is worthwhile considering why the items designed to probe algorithms, modularity and representation may have shown the highest percentage of improvers. CAL-KIBO emphasizes the relationship between CS concepts and literacy, drawing on children's stories as inspiration for programming projects and other exercises. As students engage in KIBO coding activities, they learn programming fundamentals such as recognizing the relationship between symbols on the KIBO programming blocks and concrete actions (e.g., movement of the robot). They also learn that using certain blocks results in tangible actions while others exert effects that are not as readily visible (e.g., conditionals). As children learn these coding fundamentals, the CAL-KIBO curriculum challenges them to achieve goals such as programming simulations of storybook characters and robotic re-enactments of story elements from children's literature. Participating in these challenges requires learning about symbolism (representation), decomposition of multistep processes into executable steps (modularity) and sequencing multiple steps to achieve a desired set of actions (algorithms). By recognizing patterns that repeat within and across these creative activities, children may acquire CT skills in the above-mentioned domains. Hands-on experience with creative activities may better enable students to generalize the knowledge they acquire from coding and apply it to new situations (Basawapatna et al., 2010). Another possible reason for seeing greater effects in algorithms, modularity and representation could be if *TechCheck* is inherently more sensitive to changes in these domains. Since experience with *TechCheck* in longitudinal studies is currently limited, additional studies will be needed to examine this possibility.

TechCheck's probes of hardware/software and control structures did not show differences between the CAL and No-CAL groups. There was likely a ceiling effect for these domains since close to 90% of students across the two grades responded correctly at baseline. This left little room for improvement after the coding intervention. The *TechCheck* probes for control structures involve navigating through a maze with obstacles by following a set of conditional instructions. Young children tend to have trouble learning conditionals (Elkin et al., 2014; Strawhacker & Bers, 2015) and it is possible that this concept was not fully developed in this version of the CAL-KIBO curriculum.

The change score for debugging showed a numerically greater change for the No-CAL group than the CAL group. However, the

difference was within the variance of the results. The debugging probes in this version of *TechCheck* involve correcting an unbalanced seesaw. It is possible that the children in the No-CAL group learned problem-solving through another school subject that increased their debugging skills. In interpreting all of these post-hoc, domain-specific results, it should be kept in mind that *TechCheck* is designed as a composite measure of CT across all six domains, not as a way to precisely quantify performance in individual CT domains.

At least one author has argued that transfer of CT skills to other disciplines may not occur effectively if children are introduced to problem-solving exclusively in the context of learning to code (Curzon, 2013). Curzon (2013) suggested that teaching young children programming primarily helps them develop coding-related reasoning rather than thinking skills in other domains. By introducing unplugged activities, Curzon argued that one can invoke more powerful skills of improved logical thinking and problem solving. However, participation in unplugged activities alone may not foster the development of higher reasoning skills. Thies and Vahrenhold (2012, 2013) studied the impact of *CSunplugged* in elementary and middle school children using qualitative and semi-quantitative assessments. They did not find evidence of improvements in higher-level reasoning skills and suggested that exposure to unplugged activities alone does not necessarily lead to a generalization of learning or promote the development of higher reasoning skills. While other studies have concluded that unplugged activities alone are ineffective (Black et al., 2013), some have found unplugged activities to be equally effective to coding in promoting CT (Hermans & Aivaloglou, 2017; Metin, 2020; Wohl et al., 2015). Some authors have argued that CS education is more effective when lessons include actual technology and coding in addition to unplugged activities (Bers, 2020; Huang & Looi, 2020; Thies & Vahrenhold, 2012, 2013). More research is needed to establish whether this is true and if so, to clarify the optimal approach to integrating these elements into a CS curriculum.

The importance of knowledge transfer and generalization of learning in the emergence of CT from coding education has been emphasized by several investigators (Angeli et al., 2016; Grover et al., 2015; Ioannidou et al., 2011; Repenning et al., 2015). "Near transfer" of knowledge refers to circumstances in which there is a generalization of learning sufficient to facilitate learning of new material that is similar in nature to the original learning materials. "Far transfer" refers to the adaptation of knowledge from a learned skill to help solve entirely new types of problems which may be in completely different disciplines (Reschly & Robinson-Zañartu, 2000). While the unplugged challenges in *TechCheck* were not actual coding exercises, they were selected as probes from the same domains of CT as are embodied in the CAL-KIBO coding curriculum. In this context, the improvement in unplugged problem-solving skills observed in the CAL group can be considered a form of near transfer of knowledge.

4.1. Limitations

Our initial intention was to carry out this study with kindergarten students as well as first and second graders. Although kindergarten teachers attended professional development, we were unable to implement the kindergarten CAL-KIBO curriculum due to school closure from the COVID-19 pandemic.

The No-CAL and CAL groups were not chosen at random. CAL schools were invited to participate first and No-CAL schools were added later based on having similar school level demographics to the CAL group. We did not include specific measures of SES in this study. Socioeconomic status (SES) can impact the acquisition of coding skills and CT (Google & Gallup, 2016; Scherer & Siddiq, 2019). We cannot rule out the possibility that between-group differences in SES contributed to the observed *TechCheck* outcomes.

TechCheck is a relatively new screening instrument and the version used in this study did show ceiling effects that may have reduced the magnitude of the observed outcomes, particularly in second graders. We are in the process of validating a revised version of the *TechCheck* assessment designed to more effectively discriminate a range of CT skill levels across three grades (K, 1, 2). We recognize that children's engagement in open-ended creativity and self-expression are integral to the learning and development of CT. However, *TechCheck* does not assess students in this domain owing to limitations imposed by its multiple-choice format. In the future, other more open-ended forms of assessment may be beneficial to implement in combination with *TechCheck* to get a more comprehensive picture of the child's CT development.

Although *TechCheck* successfully detected improvement of CT skills in this study, the percentage of students showing improvement was relatively small. We hope that the revisions to the assessment that have been made subsequent to this study will render it even more sensitive to change particularly in the domains that showed ceiling effects. In addition, we hope future enhancements to the coding curriculum will lead to greater improvements in CT skills after coding instruction. Due to timing restrictions with the second-grade public schools that participated, we were not able to use the full version of the CAL-KIBO curriculum that we had originally intended to implement. Our original version for second graders had an additional 24 lessons (24 h) of instruction. Future iterations of the CAL-KIBO curriculum will be extended and should allocate more time to help scaffold abstract and advanced programming concepts.

4.2. Future directions

This study has implications for the design of future coding and CT curricula for young children. Currently, many CS professional development programs for teachers focus on the syntax and implementation of programming languages rather than techniques that foster CT skill acquisition and authentic learning in other content areas (Sands et al., 2018). It may be beneficial for professional development to emphasize teaching methods that foster students' self-expression and creativity through coding to better promote young children's CT skills. Future studies comparing technically focused coding programs to curricula like CAL-KIBO that integrate literacy and creativity components can help to establish best practices for young children.

We believe it is important that these findings be extended to other cohorts and grades, as well as other curricula and educational contexts. The *TechCheck* assessment has been translated into multiple languages (i.e., Spanish, Chinese, Turkish) and is currently being

administered in diverse research and educational settings. Future studies should explore its use in children from various cultures and neuro-diverse children.

More work is needed to understand best practices for teaching CT to young children. Future longitudinal studies should compare different the effects of different coding curricula. A logical next iteration might be to compare the effects of CAL-KIBO with those of a conventional coding curriculum and one that combines coding with unplugged activities. By this approach, we can hope to learn whether young children best acquire CT when they are taught using platform-specific coding exercises, unplugged activities or a hybrid approach using both methods.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Marina Umaschi Bers co-founded the company Kinderlab that manufactures the KIBO robot and has entered into a licensing agreement with Tufts University for the technology. The research presented is not expected to have any direct financial impact. No other authors have competing interest relating to this work.

Acknowledgements

The authors would like to thank the project coordinator Angela de Mik as well as members of the DevTech research group that contributed to this project (Madhu Govind, Ziva Hassenfeld, Pat Nero, Maya Morris, Ari Lerner, and Jaclyn Tsiang) who this work would not be possible without. The authors would also like to thank the Instructional Technology Resource Technicians. Lastly, we would like to thank all of the teachers, administrators, and children involved in this study.

Appendix A. Supplementary data

Supplementary data to this article can be found online at https://doi.org/10.1016/j.compedu.2021.104222.

Funding

This work was supported by the Department of Defense Education Activity (DoDEA) “Operation: Break the Code for College and Career Readiness”. Unique Entity Identifier: “WORLDCL10”.

Credit author statement

Emily Relkin: Conceptualization, Methodology, Validation, Formal analysis, Investigation, Data curation, Writing – original draft, Writing – review & editing, Visualization, Project administration Laura de Ruiter: Conceptualization, Validation, Formal analysis, Data curation, Writing- Original Draft, Writing – review & editing, Visualization, Supervision Marina Bers: Conceptualization, Methodology, Validation, Investigation, Writing – original draft, Writing – review & editing, Supervision, Project administration.

References

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

Angeli, C., Voogt, J., Fluck, A., Webb, M., Cox, M., Malyn-Smith, J., & Zagami, J. (2016). A K-6 Computational Thinking curriculum framework: Implications for teacher knowledge. *Educational Technology & Society, 19*(3), 47–57. Retrieved from https://www.jstor.org/stable/10.2307/jeductechsoci.19.3.47.

Arfé, B., Vardanega, T., Montuori, C., & Lavanga, M. (2019). Coding in primary grades boosts children's executive functions. *Frontiers in Psychology, 10*(2713). https://doi.org/10.3389/fpsyg.2019.02713

Barr, D., Harrison, J., & Conery, L. (2011). Computational thinking: A digital age skill for everyone. *Learning and Leading with Technology, 38*(6), 20–23. Retrieved from https://id.iste.org/docs/learning-and-leading-docs/march-2011-computational-thinking-ll386.pdf.

Barrouillet, P., & Lecas, J. (1999). Mental models in conditional reasoning and working memory. *Thinking & Reasoning, 5*(4), 289–302.

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

Basawapatna, A. R., Koh, K. H., & Repenning, A. (2010). Using scalable game design to teach computer science from middle school to graduate school. In *Proceedings of the fifteenth annual conference on Innovation and technology in computer science education* (pp. 224–228). https://doi.org/10.1145/1822090.1822154. ACM.

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

Bates, D., Maechler, M., Bolker, B., & Walker, S. (2015). Fitting linear mixed-effects models using lme4. *Journal of Statistical Software, 67*(1), 1–48. https://doi.org/10.18637/jss.v067.i01

Bell, T., & Vahrenhold, J. (2018). CS unplugged—how is it used, and does it work? In H.-J. In H.-J. Böckenhauer, D. Komm, & W. Unger (Eds.), *Adventures between lower bounds and higher altitudes: Essays dedicated to Juraj Hromkovič on the occasion of his 60th birthday* (pp. 497–521). https://doi.org/10.1007/978-3-319-98355-4_29

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

Bers, M. U. (2019). Coding as Another Language: A pedagogical approach for teaching computer science in early childhood. *Journal of Computers in Education, 6*(4), 499–528. https://doi.org/10.1007/s40692-019-00147-3

Bers, M. U. (2020). *Coding as a playground: Programming and computational thinking in the early childhood classroom* (2nd ed.). New York, NY: Routledge Press.

Black, J., Brodie, J., Curzon, P., Myketiak, C., McOwan, P. W., & Meagher, L. R. (2013). Making computing interesting to school students: Teachers' perspectives. In *Proceedings of the 18th ACM conference on innovation and technology in computer science education* (pp. 255–260). Association for Computing Machinery.

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 annual meeting of the American educational research association, Vancouver, Canada, 1* p. 25). Retrieved from https://web.media.mit.edu/~kbrennan/files/Brennan_Resnick_AERA2012_CT.pdf.

Bruner, J. S., Olver, R., & Greenfield, P. (1966). *Studies in cognitive growth*. New York: Wiley.

Chen, G., Shen, J., Barth-Cohen, L., Jiang, S., Huang, X., & Eltoukhy, M. (2017). Assessing elementary students' computational thinking in everyday reasoning and robotics programming. *Computers in Education, 109*, 162–175. https://doi.org/10.1016/j.compedu.2017.03.001

Clements, D. H., Battista, M. T., & Sarama, J. (2001). Logo and Geometry. In E. Yackel (Ed.), *Journal for research in mathematics education monograph series, 10*. https://doi.org/10.2307/749924

Clements, D. H., & Gullo, D. F. (1984). Effects of computer programming on young children's cognition. *Journal of Educational Psychology, 76*(6), 1051–1058. https://doi.org/10.1037/0022-0663.76.6.1051

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

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

Curzon, P. (2013). cs4fn and computational thinking unplugged. In *Proceedings of the 8th workshop in primary and secondary computing education* (pp. 47–50). ACM. https://doi.org/10.1145/2532748.2611263.

Dienes, Z. (2014). Using Bayes to get the most out of non-significant results. *Frontiers in Psychology, 5*(781). https://doi.org/10.3389/fpsyg.2014.00781

Elkin, M., Sullivan, A., & Bers, M. U. (2014). Implementing a robotics curriculum in an early childhood Montessori classroom. *Journal of Information Technology Education: Innovations in Practice, 13*, 153–169.

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

Feldman, D. H. (2004). Piaget's stages: The unfinished symphony of cognitive development. *New Ideas in Psychology, 22*, 175–231. https://doi.org/10.1016/j.newideapsych.2004.11.005

Fields, D. A., Lui, D., & Kafai, Y. B. (2019). Teaching computational thinking with electronic textiles: Modeling iterative practices and supporting personal projects in exploring computer science. In *Computational thinking education* (pp. 279–294). Singapore: Springer. https://doi.org/10.1007/978-981-13-6528-7_16.

Flavell, J. H., Miller, P. H., & Miller, S. A. (1993). *Cognitive development* (3rd ed.). Prentice Hall. NJ.

Fraillon, J., Ainley, J., Schulz, W., Duckworth, D., & Friedman, T. (2018). *International computer and information literacy study: ICILS 2018: Technical report*.

Goldstein, J., & Flake, J. K. (2016). Towards a framework for the validation of early childhood assessment systems. *Educational Assessment, Evaluation and Accountability, 28*(3), 273–293. https://doi.org/10.1007/s11092-015-9231-8

Google for Education. (2010). *Exploring computational thinking*. Retrieved from www.google.com/edu/resources/programs/exploring-computational-thinking/index.html#!home.

Google & Gallup. (2016). *Diversity gaps in computer science: Exploring the underrepresentation of girls, blacks and hispanics*. Retrieved from https://services.google.com/fh/files/misc/diversity-gaps-in-computer-science-report.pdf.

Grover, S., Cooper, S., & Pea, R. (2014). Assessing computational learning in K-12. In *Proceedings of the 2014 conference on Innovation & technology in computer science education* (pp. 57–62). ACM. https://doi.org/10.1145/2591708.2591713.

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

Grover, S., Pea, R., & Cooper, S. (2015). Designing for deeper learning in a blended computer science course for middle school students. *Computer Science Education, 25*(2), 199–237. https://doi.org/10.1080/08993408.2015.1033142

Guzdial, M., & Morrison, B. (2016). Seeking to making computing education as available as mathematics or science education. *Communications of the ACM, 59*(11), 31–33. https://doi.org/10.1145/3000612

Hassenfeld, Z. R., Govind, M., de Ruiter, L. E., & Bers, M. U. (2020). If you can program, you can write: Learning introductory programming across literacy levels. *Journal of Information Technology Education: Research, 19*, 65–85. https://doi.org/10.28945/4509

Heckman, J., & Masterov, D. (2007). The productivity argument for investing in young children. *Review of Agricultural Economics, 29*(3), 446–493.

Hermans, F., & Aivaloglou, E. (2017). To scratch or not to scratch?: A controlled experiment comparing plugged first and unplugged first programming lessons. In *Proceedings of the 12th workshop on primary and secondary computing education* (pp. 49–56). Association for Computing Machinery.

Huang, W., & Looi, C.-K. (2020). A critical review of literature on "unplugged" pedagogies in K-12 computer science and computational thinking education. Computer Science Education. *Advance online publication*. https://doi.org/10.1080/08993408.2020.1789411

International Society for Technology in Education. (2017). *ISTE standards for students*. Retrieved from https://www.iste.org/standards/for-students.

Ioannidou, A., Bennett, V., Repenning, A., Koh, K. H., & Basawapatna, A. (2011). *Computational thinking patterns. Paper presented at the annual meeting of the American educational research association*. New Orleans, LA. Retrieved from https://files.eric.ed.gov/fulltext/ED520742.pdf.

Janveau-Brennan, G., & Markovits, H. (1999). The development of reasoning with causal conditionals. *Developmental Psychology, 35*(4), 904–911.

Jenkins, T. (2002). On the difficulty of learning to program. In *Proceedings of the 3rd annual. Conference of the LTSN centre for information and computer sciences* (pp. 53–58). Leeds, UK. Retrieved from http://www.psy.gla.ac.uk/~steve/localed/jenkins.html.

K-12 Computer Science Framework Steering Committee. (2016). *K–12 computer science framework*. Retrieved from https://k12cs.org.

Kazakoff, E. R., & Bers, M. U. (2014). Put your robot in, Put your robot out: Sequencing through programming robots in early childhood. *Journal of Educational Computing Research, 50*(4), 553–573. https://doi.org/10.2190/EC.50.4.f

Kuznetsova, A., Brockhoff, P. B., & Christensen, R. H. B. (2017). lmerTest package: Tests in linear mixed effects models. *Journal of Statistical Software, 82*(13), 1–26. https://doi.org/10.18637/jss.v082.i13

Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., Malyn-Smith, J., & Werner, L. (2011). Computational thinking for youth in practice. *ACM Inroads, 2*(1), 32–37. https://doi.org/10.1145/1929887.1929902

Lockwood, J., & Mooney, A. (2018). Computational thinking in education: Where does it fit? A systematic literary review. *International Journal of Computer Sciences and Engineering Systems, 2*(1), 41–60. https://doi.org/10.21585/ijcses.v2i1.26

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

Massachusetts Department of Elementary and Secondary Education. (2016). *Massachusetts digital literacy and computer science (DLCS) curriculum framework june 2016*. http://www.doe.mass.edu/frameworks/dlcs.pdf.

McDevitt, T. M., & Ormrod, J. E. (2002). *Child development and education*. Upper Saddle River, NJ: Merrill/Prentice Hall.

McMillan, J. H. (2013). *Classroom assessment: Principles and practice for effective instruction* (6th ed.). Boston: Pearson/Allyn and Bacon.

Metin, S. (2020). Activity-based unplugged coding during the preschool period. *International Journal of Technology and Design Education*, 1–17.

Mioduser, D., Levy, S. T., & Talis, V. (2009). Episodes to scripts to rules: Concrete-abstractions in kindergarten children's explanations of a robot's behavior. *International Journal of Technology and Design Education, 19*(1), 15–36.

Morey, R., & Rouder, J. (2018). *BayesFactor: Computation of Bayes factors for Common designs*. R package version 0.9.12-4.2 https://CRAN.R-project.org/package=BayesFactor.

Moyer, K., & Gilmer, B. V. H. (1953). The concept of attention spans in children. *Elementary School Journal, 54*(1), 464. https://doi.org/10.1086/458623

Muller, U., Overton, W. F., & Reene, K. (2001). Development of conditional reasoning: A longitudinal study. *Journal of Cognition and Development, 2*(1), 27–49.

Mullis, I. V., & Martin, M. O. (2019). *PIRLS 2021 assessment frameworks. International association for the evaluation of educational achievement. Herengracht 487*. Amsterdam, 1017 BT, The Netherlands. Retrieved from https://eric.ed.gov/?id=ED606056.

National Governors Association Center for Best Practices & Council of Chief State School Officers. (2010). *Common Core state standards*. Washington, DC. Retrieved from http://www.corestandards.org/about-the-standards/branding-guidelines/.

National Research Council. (2011). *Report of a workshop on the pedagogical aspects of computational thinking.* Washington, DC: National Academies Press.

Nouri, J., Zhang, L., Mannila, L., & Norén, E. (2020). Development of computational thinking, digital competence and 21st century skills when learning programming in K-9. *Education Inquiry, 11*(1), 1–17. https://doi.org/10.1080/20004508.2019.1627844

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

Piaget, J. (1953). *The origins of intelligence in the child.* London: Routledge and Kegan Paul.

Piaget, J. (1971). Developmental stages and developmental processes. In D. R. Green, M. P. Ford, & G. B. Flamer (Eds.), *Measurement and Piaget* (pp. 172–188). New York: McGraw-Hill.

Portelance, D. J., Strawhacker, A., & Bers, M. U. (2015). Constructing the ScratchJr programming language in the early childhood classroom. *International Journal of Technology and Design Education*. https://doi.org/10.1007/s10798-015-9325-0

R Core Team. (2019). *R: A language and environment for statistical computing. R foundation for statistical computing*. Vienna: Austria. https://www.R-project.org/.

Relkin, E., de Ruiter, L., & Bers, M. U. (2020). TechCheck: Development and validation of an unplugged assessment of computational thinking in early childhood education. *Journal of Science Education and Technology*. https://doi.org/10.1007/s10956-020-09831-x

Repenning, A., Webb, D. C., Koh, K. H., Nickerson, H., Miller, S. B., Brand, C., Her Many Horses, I., Basawapatna, A., Gluck, F., Grover, R., Gutierrez, K., & Repenning, N. (2015). Scalable game design: A strategy to bring systemic computer science education to schools through game design and simulation creation. *ACM Transactions on Computing Education, 15*(2). https://doi.org/10.1145/2700517

Reschly, D. J., & Robinson-Zañartu, C. (2000). Evaluation of aptitudes. *Handbook of Psychological Assessment*, 183–202.

Resnick, M., & Siegel, D. (2015). A different approach to coding. *International Journal of People-Oriented Programming, 4*(1), 1–4. https://doi.org/10.4018/IJPOP.2015010101

Resnick, M., & Silverman, B. (2005). Some reflections on designing construction kits for kids. In *Proceeding of the 2005 conference on interaction design and children - IDC '05* (pp. 117–122). https://doi.org/10.1145/1109540.1109556

Roman-Gonzalez, M., Moreno-Leon, J., & Robles, G. (2019). Combining assessment tools for a comprehensive evaluation of computational thinking interventions. In *Computational thinking education* (pp. 79–98). Singapore: Springer. Retrieved from https://link.springer.com/chapter/10.1007/978-981-13-6528-7_6.

Román-González, M., Pérez-González, J. C., & Jiménez-Fernández, C. (2017). Which cognitive abilities underlie computational thinking? Criterion validity of the computational thinking test. *Computers in Human Behavior, 72*, 678–691. https://doi.org/10.1016/j.chb.2016.08.047

Roman-Gonzalez, M., Perez-Gonzalez, J. C., Moreno-Leon, J., & Robles, G. (2018). Can computational talent be detected? Predictive validity of the computational thinking test. *International Journal of Child-Computer Interaction, 18*, 47–58. https://doi.org/10.1016/j.ijcci.2018.06.004

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

Sattler, J. M. (2014). In J. M. Sattler (Ed.), *Foundations of behavioral, social and clinical assessment of children.* Publisher, Incorporated.

Scherer, R., & Siddiq, F. (2019). The relation between students' socioeconomic status and ICT literacy: Findings from a meta-analysis. *Computers & Education, 138*, 13–32. https://doi.org/10.1016/j.compedu.2019.04.011

Selby, C. C., & Woollard, J. (2013). Computational thinking: The developing definition. In *Paper Presented at the 18th annual conference on innovation and Technology in Computer Science Education*. Canterbury. Retrieved from https://eprints.soton.ac.uk/356481/.

Stanovich, K. E. (1986). Matthew effects in reading: Some consequences of individual differences in the acquisition of literacy. *Reading Research Quarterly, 21*(4), 360–407. https://doi.org/10.1598/RRQ.21.4.1

Stanovich, K. E. (2000). Progress in understanding reading: Scientific foundations and new frontiers. *Guilford Press*.

Strawhacker, A. L., & Bers, M. U. (2015). "I want my robot to look for food": Comparing children's programming comprehension using tangible, graphical, and hybrid user interfaces. *International Journal of Technology and Design Education, 25*(3), 293–319. https://doi.org/10.1007/s10798-014-9287-7

Strawhacker, A. L., Lee, M. C., & Bers, M. U. (2017). Teaching tools, teachers' rules: exploring the impact of teaching styles on young children's programming knowledge in ScratchJr. *International Journal of Technology and Design Education*. https://doi.org/10.1007/s10798-017-9400-9

Strawhacker, A., & Bers, M. U. (2019). What They Learn when They Learn Coding: Investigating cognitive domains and computer programming knowledge in young children. *Educational Technology Research & Development, 67*(3), 541–575. https://doi.org/10.1007/s11423-018-9622-x

Sullivan, A., & Bers, M. U. (2015). Robotics in the early childhood classroom: Learning outcomes from an 8-week robotics curriculum in pre-kindergarten through second grade. *International Journal of Technology and Design Education*.

Sullivan, A., Elkin, M., & Bers, M. U. (2015). KIBO Robot Demo: Engaging young children in programming and engineering. Medford, MA, June 21–25. In *Proceedings of the 14th international conference on interaction design and children (IDC '15)*. New York, NY: ACM. Retrieved from https://sites.tufts.edu/devtech/files/2018/02/IDC-KIBO-Demo-Complete.pdf.

Sullivan, A., Strawhacker, A., & Bers, M. U. (2017). Dancing, drawing, and dramatic robots: Integrating robotics and the arts to teach foundational STEAM concepts to young children. In M. S. Khine (Ed.), *Robotics in STEM education: Redesigning the learning experience.* (pp. 231–260). Springer Publishing.

Tang, X., Yin, Y., Lin, Q., Hadad, R., & Zhai, X. (2020). Assessing computational thinking: A systematic review of empirical studies. *Computers in Education, 148* (103798). https://doi.org/10.1016/j.compedu.2019.103798

Thies, R., & Vahrenhold, J. (2012). Reflections on outreach programs in CS classes: Learning objectives for" unplugged" activities. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education* (pp. 487–492). https://doi.org/10.1007/978-3-319-98355-4_29

Thies, R., & Vahrenhold, J. (2013). *On plugging unplugged into CS classes*. https://doi.org/10.1145/2445196.2445303

Virginia Department of Education. (2017). *VDOE: Computer science standards of learning*. retrieved from http://www.doe.virginia.gov/testing/sol/standards_docs/computer-science/index.shtml.

Werner, L., Denner, J., & Campe, S. (2014). Using computer game programming to teach computational thinking skills. Learning. *Education And Games, 37*. Retrieved from https://dl.acm.org/citation.cfm?id=2811150.

Werner, L., Denner, J., Campe, S., & Kawamoto, D. C. (2012). The fairy performance assessment: Measuring computational thinking in middle school. *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education*, 215–220. https://doi.org/10.1145/2157136.2157200

Wetzels, R., Matzke, D., Lee, M. D., Rouder, J. N., Iverson, G. J., & Wagenmakers, E. J. (2011). Statistical evidence in experimental psychology: An empirical comparison using 855 t tests. *Perspectives on Psychological Science, 6*(3), 291–298. https://doi.org/10.1177/1745691611406923

Wing, J. (2006). Computational thinking. *CACM, 49*(3), 33–36. https://doi.org/10.1145/1118178.1118215. March 2006.

Wing, J. (2011). *Research notebook: Computational thinking—what and why? The link magazine*. Pittsburgh: Spring. Carnegie Mellon University. Retrieved from https://www.cs.cmu.edu/link/research-notebookcomputational-thinking-what-and-why.

Wohl, B., Porter, B., & Clinch, S. (2015). Teaching computer science to 5–7 year-olds: An initial study with scratch, cubelets and unplugged computing. In *Proceedings of the workshop in primary and secondary computing education* (pp. 55–60).

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

Zapata-Cáceres, M., Martín-Barroso, E., & Román-González, M. (2020). Computational thinking test for Beginners: Design and content validation. In *2020 IEEE global engineering education conference (EDUCON)* (pp. 1905–1914). IEEE. https://doi.org/10.1109/EDUCON45650.2020.9125368.

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