Early Childhood Education Journal
https://doi.org/10.1007/s10643-026-02252-2

Do Early CS Skills Transfer? Investigating the Academic Impact of Coding and CT in K–2 Classrooms

Ghaida S. Alrawashdeh[1,2] · **Marina U. Bers**[3]

Received: 15 May 2025 / Accepted: 9 May 2026

Abstract
While there is growing interest in introducing computer science (CS) education in early grades, causal evidence of its impact on both CS-specific outcomes like coding and computational thinking (CT) and broader academic skills such as math and literacy remains limited and mixed. This study evaluates a randomized controlled trial of a CS curriculum introduced to K-2 students across two U.S. states. A total of 1,705 students participated, with the treatment group receiving the CS curriculum and the delayed-treatment group continuing business-as-usual instruction. Multilevel modeling assessed the overall intervention effects. A path analysis examined the relationship between gains in coding and CT skills and improvements in math and literacy. Results showed that students in the treatment group demonstrated significant gains in coding and CT compared to controls. Growth in coding skills significantly predicted improvements in both math and literacy outcomes. These findings underscore the potential of early CS education to support foundational academic skills, particularly when curricula are inclusive and intentionally integrated. Policy implications include the need for early and equitable access to CS instruction, professional development for educators, and alignment of CS content with broader academic goals to maximize its educational value.

Keywords Computer science · Early childhood education · Coding · Computational thinking · Math · Literacy · Disparities

Introduction

Emerging trends in early childhood education highlight a departure from traditional literacy and numeracy focuses to integrate coding and computational thinking (CT) as foundational competencies (Angeli & Giannakos, 2020; Kafai & Burke, 2014; Tikva & Tambouris, 2021; Wing, 2006). This shift reflects a growing recognition of the essential role these skills play in the digital age (Chen et al., 2017; Kite et al., 2021). CT, as conceptualized by Jeannette Wing (2006, 2019), involves the cognitive processes of framing problems and designing solutions that can be implemented by computers, humans, or machines.

Unlike coding, which focuses on translating human-readable instructions into executable computer commands (Egbert et al., 2021), CT extends beyond computer science (CS) to encompass analytical and problem-solving skills applicable across disciplines such as math and writing (Bers et al., 2022; Li et al., 2022; Wing, 2019). These abilities enable individuals to break down complex problems, develop systematic solutions, and critically evaluate their effectiveness, a skill set crucial for success in diverse domains (Seoane Pardo, 2018; Yadav et al., 2011).

This study examines whether early exposure to coding and computational thinking supports learning beyond computer science, specifically in core academic domains such as math and literacy. In doing so, it builds on prior

✉ Ghaida S. Alrawashdeh
galrawashdeh@oakland.edu

Marina U. Bers
marina.bers@bc.edu

[1] Department of Teaching and Learning at School of Education and Human Services, Oakland University, Rochester, MI 48309-448, USA

[2] Technology-Enhanced Learning & Teaching Research Group, School of Education and Human Services, Oakland University, Rochester, MI 48309-448, USA

[3] DevTech Research Group, The Lynch School of Education and Human Development, Boston College, MA 02467140 Commonwealth Avenue, Chestnut Hill, USA

Published online: 28 May 2026

work evaluating the Coding as Another Language (CAL) curriculum and extends this line of research by focusing on the question of skill transfer. We adopt the following definitions: Computer Science (CS) is the academic discipline encompassing the study of computers and algorithmic processes. Computational Thinking (CT) refers to the broader cognitive and problem-solving skills (e.g., decomposition, pattern recognition, abstraction) that are foundational to CS but applicable across domains. Coding (or programming) is the practical act of translating instructions into a language a computer can execute, serving as a primary vehicle for developing CT skills in early childhood contexts.

Childhood represents a critical developmental window for cultivating these foundational skills (Wing, 2008). Early integration not only equips students for a digital future but also empowers them as creators and innovators rather than mere consumers of technology (Chen et al., 2017). This developmental approach aligns with theories emphasizing children's cognitive progression from concrete to abstract thinking during this formative period (Bati, 2022; Piaget, 1973). However, achieving mastery in these skills necessitates deliberate pedagogical approaches, expert guidance, and dedicated instructional time (Atmatzidou & Demetriadis, 2016; Cachero et al., 2020; Kite et al., 2021).

In response, a global movement has emerged to integrate CS and CT into early elementary education (Angeli & Giannakos, 2020; Bers et al., 2022; Egbert et al., 2021; Özcan et al., 2021; Papadakis, 2024; Tikva & Tambouris, 2021; Upadhyaya et al., 2020; Zhang & Nouri, 2019). Initiatives such as Europe's "Informatics for All" coalition and the International Society for Technology in Education's computational thinking competencies reflect efforts to support systematic integration across disciplines. In the United States, this momentum is evident in the inclusion of CT within the Next Generation Science Standards and in funding initiatives from agencies such as the National Science Foundation and the Department of Education. Together, these developments signal a broader recognition of CS as a means of fostering critical thinking, problem-solving, and adaptability (Durak & Saritepeci, 2018).

Although previous studies have examined the growing role of CS in K–12 education (Barcelos et al., 2018; Grover & Pea, 2013; Hickmott et al., 2018; Hsu et al., 2018; Tang et al., 2020), it remains unclear whether the skills students develop through coding and CT meaningfully support learning in other subjects like math and literacy (Lei et al., 2020; Miller, 2019; Ye et al., 2023). Existing findings are mixed, with some studies showing benefits, particularly for elementary students (Cheng et al., 2023; Kaup et al., 2023; Lei et al., 2020), and others showing limited impact, especially in under-resourced schools (Doleck et al., 2017; Hu & Wang, 2024). Furthermore, a lack of research in East Coast states highlights the need for more regionally diverse studies on skill transfer (Upadhyaya et al., 2020). Persistent gaps in disaggregated data by race and socioeconomic status further limit our understanding of equity in computing education outcomes (Upadhyaya et al., 2020).

Building on these gaps, the present study investigates the impact of introducing the equity-focused CAL curriculum on K-2 students. CAL uses strategies such as block-based programming to reduce literacy barriers and incorporates diverse role models to promote inclusion. This study extends a previously published evaluation of CAL's first-year implementation (masked citation) in two key ways. First, it incorporates data from both years of a two-phase randomized controlled trial (RCT), enabling analysis of sustainability and the effects of shifting professional development models (see Methods). Second, it examines whether gains in coding and computational thinking skills transfer to math and literacy outcomes. This investigation was guided by the following research questions:

RQ1. To what extent does participation in a CS curriculum lead to growth in students' coding and computational thinking skills in early grades?
RQ2. Is there a relationship between students' improvement in coding and computational thinking skills and their literacy and math outcomes?

Literature Review

Computational Thinking and Coding: Conceptual Foundations

Seymour Papert's seminal book *Mindstorms: Children, Computers, and Powerful Ideas* (1993) laid the groundwork for the integration of CS into early childhood education, particularly through programming languages like LOGO. Papert argued that computers should not be seen as mere tools but as transformative instruments capable of cultivating critical thinking, creativity, and procedural skills in young learners. His vision introduced the concept of children interacting with computers as "objects-to-think-with," promoting active exploration and knowledge construction. He highlighted programming as a concrete framework for developing effective problem-solving strategies and enriching children's understanding through direct engagement with computational concepts. Papert emphasized hands-on programming to enhance problem-solving abilities and deepen understanding of computational concepts.

This perspective set the stage for the formalization of CT in modern education by Jeannette Wing (2006, 2008, 2019). Distinguishing it from traditional CS, Wing emphasized CT

as essential for problem-solving across diverse disciplines. She defined CT as the ability to conceptualize problems, formulate solutions, and evaluate their effectiveness, emphasizing creativity and abstraction over rote programming skills (Weintrop et al., 2016; Wing, 2006). Despite ongoing debates over a definitive CT framework (Moreno-León et al., 2018; Shute et al., 2017), current educational strategies focus on key CT practices like pattern recognition, problem decomposition, abstraction creation, algorithm development, and debugging (Grover & Pea, 2013; Shute et al., 2017). Yadav et al. (2011) and Selby et al. (2014) expand this discourse to include automation, evaluation, and generalization. Fagerlund et al. (2021) conclude that CT encompasses a spectrum of cognitive tasks essential for navigating the computational world and employing multidimensional problem-solving skills across diverse domains.

For that, Wing (2019) posits it as a foundational skill set for navigating the complexities of today's digital landscape, advocating for its role in cultivating adaptive thinking and innovative problem-solving strategies. Central to Wing's argument is the assertion that CT precedes programming in problem-solving and that it can occur independently, underscoring its broader applicability beyond coding contexts (Wing, 2006, 2019). In contrast, Papert (1993) contends that while learning strategies can be developed independently of computers, programming provides a tangible model of thought that enhances understanding of problem-solving methodologies. By conceptualizing the computer as an 'object-to-think-with,' Papert argues, children engage with a dynamic microworld, thereby moving beyond mechanical approaches and fostering a proactive, self-directed learning process.

Current research highlights coding as a cognitive pathway that supports core CT processes, showing significant correlations between coding proficiency and improved CT abilities (Egbert et al., 2021; Sun et al., 2022; Tikva & Tambouris, 2021; Yang & Lin, 2024). For instance, Tondeur et al. (2019) conducted a meta-analysis revealing substantial positive impacts of coding education on both technical skills ($\bar{g}$ = 0.75) and broader cognitive competencies ($\bar{g}$ = 0.47), aligning with Papert's assertions regarding the holistic benefits of programming experiences. However, some scholars caution against an exclusive focus on coding, arguing that it may inadvertently limit the scope of CT to binary code and neglect its interdisciplinary potential (Haseski et al., 2018; Sun et al., 2022; Wing, 2019). Such concerns underscore the importance of cultivating CT skills in contexts beyond programming, ensuring a holistic approach that harnesses CT's full spectrum of applications across diverse fields.

Also, in the context of early childhood education, the terms programming and coding are often used interchangeably, despite their distinct processes (Zhang & Nouri, 2019). In our study, we adopt an interchangeable usage of these terms to reflect their common application in early childhood settings, where the complexity of full programming tasks may exceed developmental capabilities. To clarify their relationship, we provide a visualization in Fig. 1.

Figure 1 illustrates the interrelated nature of CT, CS, and coding. Rather than a strict hierarchy, we conceptualize these as overlapping domains with bidirectional influences (Wing, 2006). CT represents the broadest set of cognitive irategesst (Lai & Ellefson, 2023). CS applies these strategies within a specific disciplinary context. Coding is a concrete, situated practice within CS that actively engages CT skills. The arrows indicate bi-directional influence: engaging in coding develops CT, and stronger CT facilitates more sophisticated coding and CS understanding.

In early childhood education research, coding activities serve as the primary, observable context for developing and studying CT. The following section reviews empirical evidence on how coding instruction (and the CT processes it engages) relates to learning in math and literacy. The following section reviews this body of work, examining how the concrete practice of coding, and by extension, the engagement of underlying CT processes, relates to learning in math and literacy.

CS and Academic Development in Math and Literacy

Core CS practices such as abstraction, algorithmic thinking, problem-solving, and systematic reasoning are not only essential to coding but are also transferable to other academic domains, notably math and literacy (Grover & Pea, 2013; Lei et al., 2020). Both math and literacy are fundamental to children's academic success and lifelong learning, yet students often encounter persistent challenges in mastering them (Claessens & Engel, 2013; Fuchs et al., 2021). This has prompted growing interest in using coding and CT as a bridge to strengthen children's broader academic development.

The relationship between CT and academic achievement appears stronger in subjects closely aligned with CT, such as math and science, compared to more distantly related subjects (Kastberg et al., 2015). For instance, studies have reported notably higher correlations between CT and math achievement (α=0.74) than between CT and history achievement (α=0.17; Korkmaz, 2012; Limiñana Gras et al., 2010). Gülmez and Özdener (2015) further found that students' CT skills were significantly linked to academic achievement in mathematics, information technology, and in languages, namely Turkish and English.

In mathematics, CT processes support key skills such as identifying patterns, building systematic procedures, and understanding abstract relationships. As Lei et al. (2020)

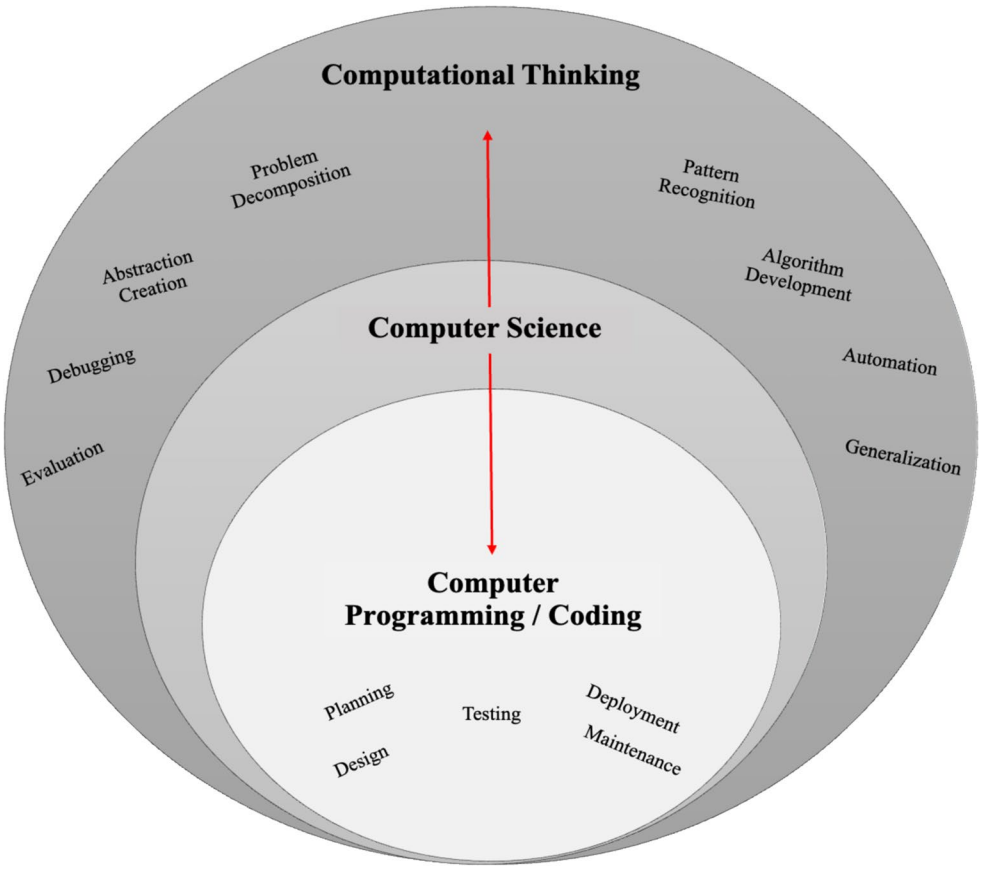

Fig. 1 Relationship between computational thinking, coding and programming, depicting various tasks (not Exhaustive)

note in their review, Grover and Pea (2013) emphasize that CT mirrors many mathematical operations: students' abstract numbers from real-world quantities, apply algorithms to solve arithmetic problems, and build systems of relationships between mathematical concepts. Recognizing these links, Papert (1993) proposed that coding could serve as an entry point for math education, suggesting that programming environments like Logo could make abstract concepts more concrete and intuitive. Instead of working with mathematical concepts in isolation, children use them to build, test, and refine creative projects, deepening their understanding through meaningful application (Bers, 2019; Papert, 1993).

A growing body of evidence supports the view that coding and CT promote mathematical development. Early studies found that programming with Logo enhanced students' problem-solving abilities and geometric thinking (Battista & Clements, 1986). More recent research has demonstrated that coding instruction improves mathematical reasoning, self-efficacy, spatial awareness, and overall math aptitude across age groups (Messer et al., 2018; Psycharis & Kallia, 2017; Salac et al., 2021). Several large-scale evaluations reported significant gains in math performance following CS instruction, including a $\beta = 19.88$ ($p < .05$, $n = 5{,}791$) in Century et al. (2020) and a partial eta-squared of 0.0625 in Salac et al. (2021). Additional findings, such as improvements in coordinate graphing and visual–spatial skills (Friend et al., 2018; Lewis et al., 2015), suggest that problem-solving-oriented CS activities enhance mathematical reasoning.

However, not all initiatives have yielded positive outcomes. Dohn (2020) found that a poorly designed coding curriculum led to declines in students' motivation toward both coding and mathematics, underscoring the need for carefully designed programs that promote exploration, creativity, and autonomy, supported by sustained professional development.

Although evidence supporting the impact of coding on literacy outcomes is growing, findings are more mixed. Some studies observed positive effects, such as Hassenfeld et al. (2020), who reported a moderate positive correlation between CS programming and phonological awareness ($r = .30$, Bayes factor = 43.85), and Salac et al. (2020), who found statistically significant differences based on reading proficiency. Other studies, including Century et al. (2020) and Salac et al. (2021) did not find significant effects on English Language Arts or reading assessments. Overall, the evidence suggests a stronger and more consistent link between CS education and math performance than with literacy outcomes, though variation in instructional approaches and assessment measures likely contributes to the mixed results.

Nevertheless, coding offers clear parallels with literacy development. Bers (2017) frames coding as an extension of literacy rather than a separate technical skill. Programming languages, like spoken and written languages, require mastery of syntax, grammar, and coherent structure (Vee, 2017). Lei et al. (2020) note that in language arts, students' abstract words and grammar rules to construct coherent sentences, applying algorithmic structures such as noun–verb–direct object patterns.

Classroom studies show that integrating coding can boost writing quality, writing stamina, and students' confidence in their writing abilities (Thompson & Childers, 2021). Learning to code also fosters new forms of creative expression. Multimodal projects using Scratch have helped children develop storytelling skills by combining algorithmic design with visual and structural composition (Whyte et al., 2020). Coding activities also encourage planning, organizing, and logically sequencing ideas, which are critical skills for reading comprehension and writing.

Building on this foundation, the present study introduces the CAL curriculum, designed to integrate computational concepts and literacy skills for K-2 students. The CAL curriculum combines plugged and unplugged (divorced from a programming language) activities to encourage storytelling, free expression, and playful exploration. Additionally, this trial included a two-phase PD model. In the first year, teachers were trained by an expert and some participants received further training at a higher education institute. These teachers then led PD sessions in the second year, promoting sustainability and empowering educators to lead CT integration efforts within their own classrooms.

Methods

In this cluster-randomized study, we engaged 1,705 K-2 students from 31 elementary schools across two East Coast states, with 18 schools in State 'A' and 13 in State 'B'. As mentioned earlier, there is limited research on K-12 CS education in these states (Upadhyaya et al., 2020). The student body consisted of 458 kindergarteners, 561 first graders, and 686 s graders (total $n = 1{,}705$; see Table 1). Students in grades K-2 ranged in age from 5 to 10 years old, with an average age of 7.2 years ($SD = 1.05$). Grades K-2 were selected as they represent a critical, sequential period for establishing foundational skills in both core academics and computational thinking, allowing us to examine developmental progression within early childhood (Bers, 2019; Wing, 2019). The intervention spanned from 2021 to 2023. Schools were randomized to treatment or control groups. The treatment group received the CAL curriculum in SY2021-2022 (Phase I), while the control group followed the regular curriculum and received the CAL curriculum in SY2022-2023 (Phase II).

It is important to note that during Phase I, the control group followed business as usual. While both states' standards mandate the teaching of coding and CT, this typically occurred through integration into other subjects without dedicated instructional time or a formal, standalone curriculum. This ensured that control students were still exposed to foundational CS concepts, though in a less structured manner. The CAL intervention, in contrast, represented a shift from this integrated exposure to deliberate, scaffolded CS instruction.

Randomization occurred at the school level to ensure uniform treatment assignment across classrooms within each school. In one state, schools were randomized within three strata based on poverty quartile, while in the other

Table 1 Demographic profile and learning needs diversity in the recruited sample with available information ($n = 1{,}705$)

	Condition	
Variable	Treatment	Control
Cohort		
Year 1	619 (54.06%)	560 (100.00%)
Year 2	526 (45.94%)	0 (0.00%)
Gender		
Assigned Female at Birth (AFAB)	580 (50.66%)	291 (51.96%)
Assigned Male at Birth (AMAB)	565 (49.34%)	269 (48.04%)
Grade		
Grade 2	444 (38.78%)	242 (43.21%)
Grade 1	347 (30.31%)	214 (38.21%)
K	354 (30.92%)	104 (18.57%)
Race/Ethnicity		
White	632 (59.79%)	368 (65.71%)
Hispanic	204 (19.30%)	83 (14.82%)
Asian	51 (4.82%)	17 (3.04%)
Black or African American	107 (10.12%)	65 (11.61%)
Mixed or Other	56 (5.30%)	25 (4.46%)
American Indian or Alaska Native	7 (0.66%)	2 (0.36%)
Site		
State A	380 (33.19%)	82 (14.64%)
State B	765 (66.81%)	478 (85.36%)
Individualized Educational Plan (IEP)		
No	973 (84.98%)	492 (87.86%)
Yes	172 (15.02%)	68 (12.14%)
English Language Learner (ELL)		
No	950 (82.97%)	505 (90.18%)
Yes	195 (17.03%)	55 (9.82%)
Socioeconomic status (SES)		
High	775 (67.69%)	426 (76.07%)
Low	370 (32.31%)	134 (23.93%)

state, schools were randomized within three strata based on the number of participating classrooms across three grade levels, as determined by state officials. As a result, the representation of certain student groups, such as those from low-income backgrounds, mirrors the actual population distribution in these districts. Therefore, while the sample may not include a larger proportion of specific student groups, it is consistent with the typical composition of the student body in these regions.

A total of 33 schools were initially randomized. Of these, 15 schools (9 in State A, 6 in State B) were assigned to receive the CAL curriculum in the first year, and 17 schools (10 in State A, 7 in State B) were allocated to the delayed-treatment condition. Prior to implementation, 2 treatment schools and 3 control schools in State A dropped out due to COVID-19 constraints. One control school was replaced, resulting in 13 active treatment schools and 15 active control schools. Data were collected from 28 schools (13 treatment, 15 control) comprising 59 classrooms. The study was approved by the Institutional Review Board at Boston College, [protocol number #23.063.01], and consent forms were collected from all students in the analytical sample by site coordinators.

The CAL Curriculum

The intervention consisted of the Coding as Another Language (CAL) curriculum, a developmentally appropriate, equity-focused CS curriculum designed for K-2 students (Bers, 2019). It addresses the need for early childhood CS curricula (Upadhyaya et al., 2020) by immersing students in systems thinking and collaborative problem-solving within meaningful, literacy-rich contexts (Kite et al., 2021). The CAL curriculum addresses this need by introducing K-2 students to CS in a developmentally appropriate way, encouraging exploration, experimentation, and collaborative problem-solving with real-world challenges.

The CAL curriculum is structured around six powerful ideas that bridge CS and literacy domains: Algorithms, Modularity, Control Structures, Representation, Hardware/Software, and Design Process (see Table A1 in the Appendix for full descriptions and examples). These ideas are operationalized through 24 sequenced lessons (≈45 min each) that blend unplugged activities (focused on CT concept introduction without devices) with plugged activities on the ScratchJr platform (where students apply these concepts through coding).

The CAL curriculum offers approximately 18 h of instructional content, allowing for adaptability across diverse educational settings[1]. It is also aligned with academic frameworks, ensuring a seamless integration of coding instruction with core academic subjects. Table A2 in the appendix provides examples of how lessons in the Kindergarten CAL curriculum are aligned with particular K standards and frameworks.

Pedagogical Approach and Differentiation

CAL uses a constructivist, play-based approach where students learn through creating animated stories and games. The curriculum is differentiated by grade level to align with developmental progressions. For example, in foundational lessons on algorithms, Kindergarten focuses on oral language and physical sequencing, Grade 1 introduces symbolic representation and written procedures, and Grade 2 builds analytical skills for decomposing and specifying sequences. This scaffolding ensures that activities match students' cognitive and motor skill development (see Bers, 2019, for full developmental rationale).

This differentiation is embedded in the structure of the lessons themselves. A review of sample lessons reveals a clear, spiraling progression:

- Kindergarten establishes concrete understanding through oral language and physical action, using activities like "Word Scramble" and "Program the Teacher."
- First Grade builds abstraction by introducing symbolic representation with ScratchJr blocks and shifting students to write procedural sequences.
- Second Grade advances analytical thinking by having students label sequenced steps (e.g., first, second, third, last in journals) and explicitly contrast human and computer instructions.

This intentional scaffolding is delivered through consistent, multimodal routines and a gradual release of responsibility, ensuring all learners access computational thinking concepts through developmentally appropriate practice.

Example Lesson Structure

A representative Grade 1 lesson, "Order Matters" (Fig. 2), demonstrates the curriculum's pedagogical flow. It begins with unplugged activities targeting CT concepts: students unscramble a sentence to explore sequence, then define an algorithm. This transitions to a plugged activity where students predict and create block sequences in ScratchJr, applying the concept to build coding proficiency. The lesson concludes with a literacy-connected "How-To Book" writing

[1] Some students may require additional support, such as further division of activities or extended exploration time for each concept.

Fig. 2 A visual overview of Lesson 1 in Grade 1 CAL Curriculum

exercise. This scaffolded structure, concept (unplugged/CT) → application (plugged/coding) → creation and literacy integration, exemplifies CAL's approach to developing CT skills through coding. For complete lesson plans, scope and sequence, and standards alignments, see the curriculum manual (Bers, 2019) and Appendix Tables A1–A2.

How CAL Differs from Standard CS Exposure

Unlike the control condition's standards-based integration of CS concepts (which lacked dedicated time and a structured scope), the CAL intervention provided: (1) dedicated weekly instructional time (≈1 h/week over 6 months); (2) a structured, sequential curriculum with explicit learning objectives; (3) systematic integration of CT and literacy via the Coding as Another Language pedagogy; and (4) targeted equity strategies, including block-based programming (ScratchJr) to lower literacy barriers and diverse role model representation. Demographic variables (e.g., race, SES) are included in our analytic models; detailed equity-focused analyses are presented in the appendix.

Implementation Support

Teachers received PD training tailored to the curriculum. In Year 1, PD was expert-led. In Year 2, a peer-led model was implemented to support sustainability and local capacity building. This peer-led PD was facilitated by teachers who had already completed the expert-led training and successfully implemented the CAL curriculum in Year 1. Throughout both years, the research team provided ongoing consultation and monitored implementation fidelity (more on this in the Procedures and Fidelity section). All PD, whether expert- or peer-led, covered CAL's pedagogical approach, the six powerful ideas, and facilitation of open-ended projects (for PD details, see Alrawashdeh et al., 2024, 2025).

Assessments

Two validated assessments were administered to measure students coding and CT skills: the Coding Stages

Assessment (CSA; de Ruiter & Bers, 2022) for coding skills (reported reliability: Guttman's λ 6 = 0.94) and TechCheck (Relkin et al., 2020) for CT skills (reported reliability: α = 0.68). The Coding Stages Assessment (CSA) is a validated, performance-based measure of a child's progression through developmental stages of learning the ScratchJr programming language (de Ruiter & Bers, 2022). The TechCheck assessment is a validated, unplugged (non-digital) multiple-choice measure of core computational thinking concepts, designed as puzzle-like challenges analogous to programming problems (Relkin et al., 2020).

To administer the assessments, 54 independent research assistants, trained separately, and blinded to the study design, conducted one-on-one Zoom assessments. An external evaluation group, which was involved in supervising and monitoring the study design, data collection and evaluation of the intervention, collaborated with site coordinators to ensure unbiased selection of these assistants. Fleiss' Kappa was calculated to assess the inter-rater reliability among the assistants for a random set of 5 questions. The Kappa value indicated substantial agreement between the assistants' judgments, $\kappa = 0.656$, $z = 132$, $p < .001$.

Students' literacy and math performance was measured using one of four standardized assessments: Fastbridge, STAR, Aimsweb, or iReady. These assessments were selected and implemented independently by each school district as part of their regular benchmarking practices during the fall and spring. The research team requested and obtained the assessment data directly from the districts. Each tool evaluated key domains in early literacy and numeracy, such as phonological awareness, vocabulary, comprehension, number sense, and algebraic thinking. For kindergarten students, early literacy measures focused on phonological awareness, letter recognition, and foundational numeracy skills. First-grade assessments emphasized phonemic awareness, sight word recognition, and number sense. Second-grade assessments included reading comprehension, fluency, and multi-digit arithmetic. This grade-level differentiation ensured that assessments were appropriately calibrated to students' developmental levels.

Math and literacy assessments were administered twice annually; once in the fall (pre-intervention for treatment students) and once in the spring (post-intervention). This timing allowed for measurement of student growth across approximately 6 months of instruction. Schools administered these assessments as part of their standard benchmarking protocols, ensuring authentic assessment conditions and reducing assessment burden on classrooms. For more detailed descriptions of these assessments, please see the appendix. To account for variation across assessment types, all scores were converted to standardized z scores for analysis (Jain et al., 2005).

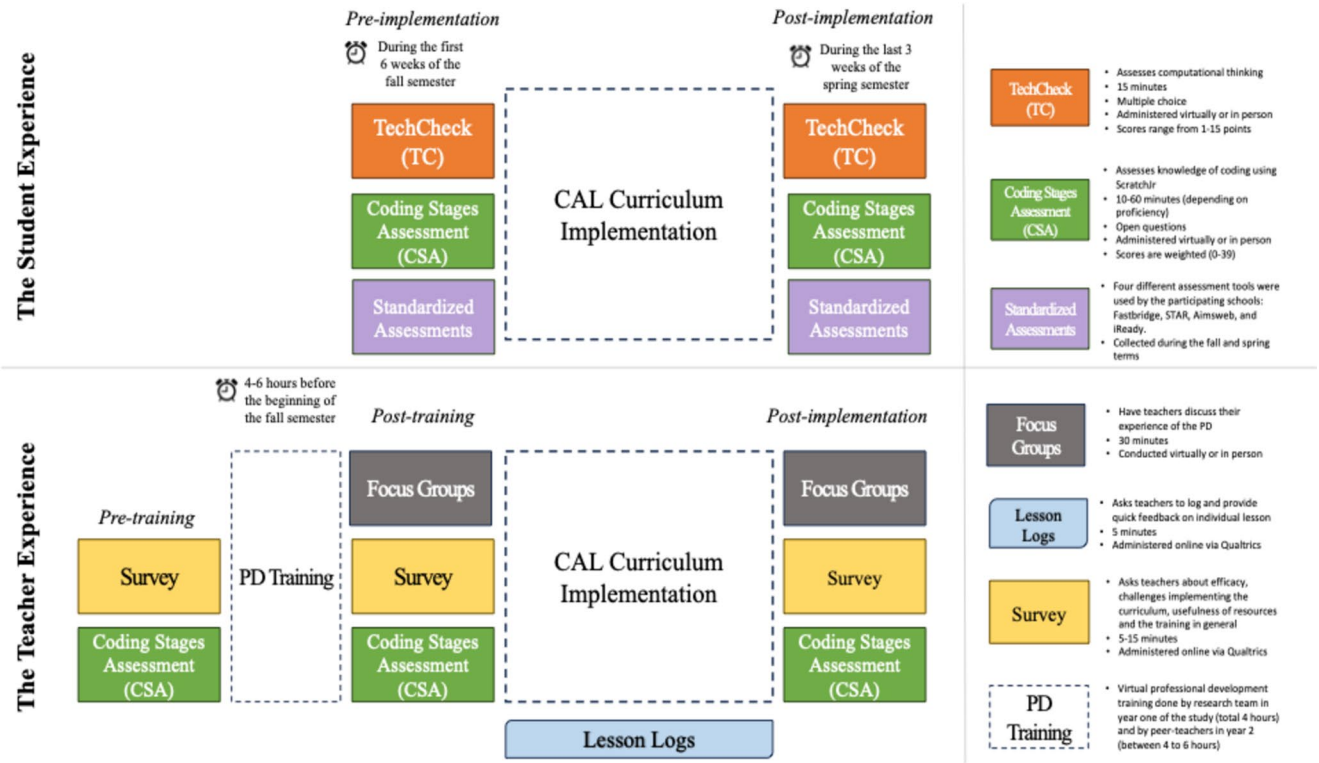

Fig. 3 An overview of the study's design, data collection instruments and timeline. *Note*. This study focuses on the student's experience. For detail regarding the teacher experience see (Alrawashdeh et al., 2024, 2025)

Procedures and Fidelity

In the first year, treatment group teachers received PD training from the research team. In the second year, educators selected by their school districts underwent specialized early childhood technology training through a certificate program at a university in New England. Most of these educators were part of the teachers who implemented the CAL curriculum the first year. Figure 3 shows the data collection instruments used in this intervention for both teachers and students. The top panel of Fig. 3 displays the sequence of student assessments before and after curriculum implementation, while the right panel describes the instruments. The bottom panel outlines the sequence of actions for teachers. Details regarding teacher training are described in (Alrawashdeh et al., 2024, 2025). Most trained teachers were classroom teachers (≈60%), with support teachers, including STEM and math specialist coaches, comprising 8%, and inclusion teachers accounting for 2%.

It is important to note that State A's intervention started two months later than State B due to administrative reasons. Consequently, most classrooms in State B completed the entire curriculum, while those in State A completed an average of 13 lessons (ranging from 7 to 20 lessons; approximately 585 min of CS). In our analysis, we used the last reported lesson taught by teachers as a proxy for dosage in the Treatment condition since precise information on the start or duration of teaching was unavailable. The intervention lasted approximately six months each year, with an average of one hour per week.

Due to pandemic-related constraints, classroom visits were not feasible, so self-reported information from teachers was used to assess the fidelity of curriculum implementation. Fidelity monitoring involved: (a) analyzing lesson logs submitted by teachers via online Qualtrics surveys after each lesson; (b) conducting post-training and post-implementation focus group discussions with teachers and site-level administrators; and (c) holding weekly meetings with the external evaluation team and site-level administrators.

The delivery mode of the CAL lessons followed district policies during the pandemic. During the Fall of 2020, instruction occurred under hybrid or remote models as schools phased their reopenings. From Spring 2021 onward, the majority of participating classrooms resumed in-person instruction, with lessons delivered in-person accordingly. Teachers adapted lesson delivery for remote or hybrid models when necessary (e.g., via Zoom). All student assessments (CSA, TechCheck) were administered one-on-one via Zoom throughout the study to ensure consistency and safety.

Detailed descriptions of the ScratchJr platform features, the specific coding and computational thinking skills targeted in each CAL lesson, and the full technical specifications and sample items for the CSA, TechCheck, and standardized academic assessments are provided in the Appendix.

Analytic Sample

Figure 4 provides the flow of participants through each stage of the intervention using the Consolidated Standards of Reporting Trials (CONSORT) diagram. We compared the mean scores of students who remained in the study with those who dropped out (4% of students in Y1 & 10% in Y2) based on their baseline results. The analysis revealed no significant difference for either the coding or CT skills scores ($p = .674$ and $p = .833$ respectively). Attrition analysis is presented in the appendix.

The analytic sample size varied by model due to data availability. Out of the 1,705 recruited students, 1,692 completed the baseline coding assessment and 1,683 completed the baseline CT assessment. Analyses of transfer to math and literacy were limited to 1,049 students, based on the standardized assessment data received from participating districts.

Also, given that the assignment was distinct; we only collected data from students who were newly assigned to the treatment in Year 2. We did not collect any scores from students who were in the control group in Year 1 and received the treatment in Year 2. The analytical sample reflects a varied student population, with a balanced representation in terms of gender (51% AFAB). Table 1 summarizes the demographic profile and diversity of learning needs in the subset of the analytic pool for whom full demographic information was available, while frequencies and percentages grouped by Condition are presented in Table A4 in the appendix.

Analysis

To address the study's first question and the cluster-randomized design, we employed a two-level multilevel model (MLM) with students as Level 1 and teachers as Level 2. MLM is commonly used to assess treatment effects on student performance, effectively analyzing outcome differences while controlling for covariates across multiple levels (Goldstein, 2011). In this model, baseline scores acted as covariates, representing predictors of future performance (Bloom et al., 2007). Separate models were developed for Coding and CT. Details regarding the model notation, centering and assumptions are provided in the appendix. The analytic sample for the multilevel models reflects all students with the necessary outcome and primary predictor data.

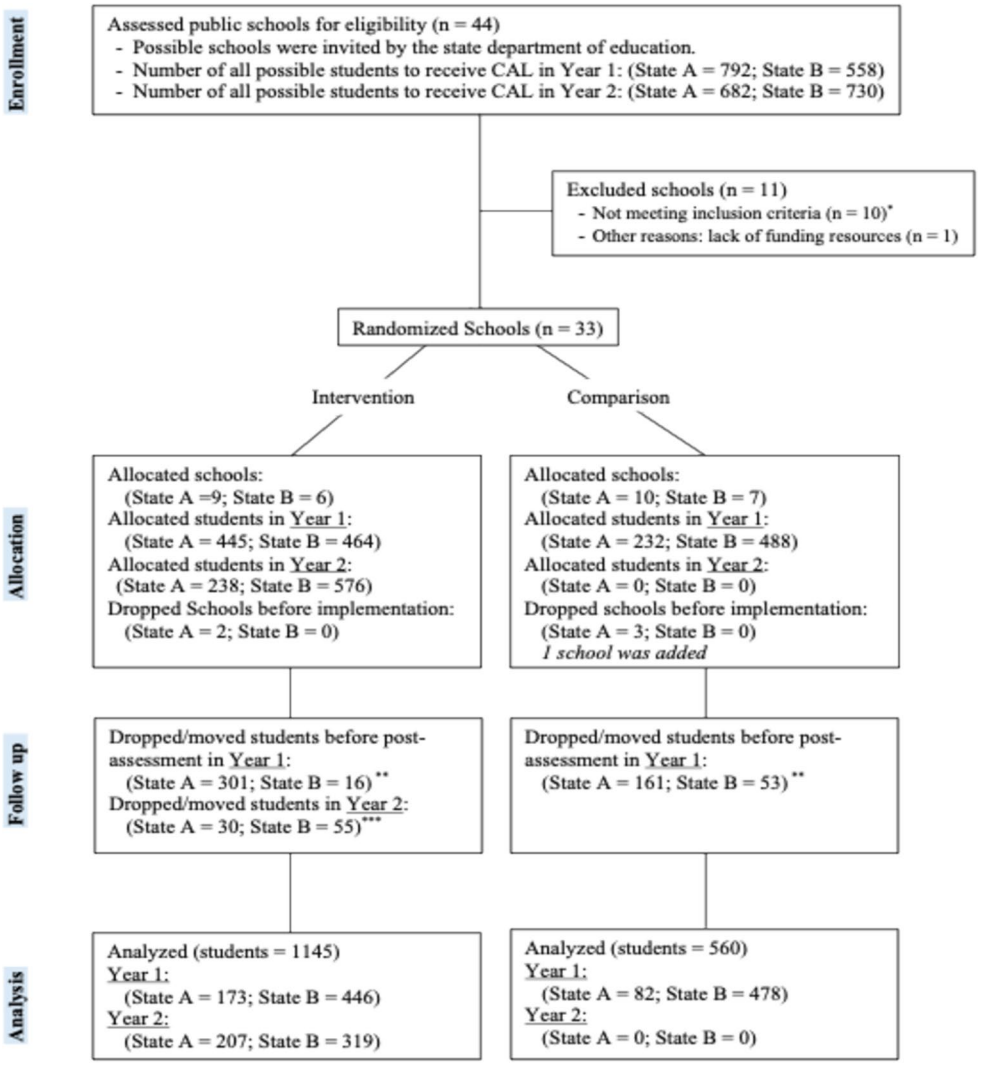

Fig. 4 CONSORT diagram showing the flow of participants through each stage of the intervention

Our analysis involved both unconditional and conditional models to estimate the average treatment effect of the intervention. First, we ran an intercept model, containing only the intercept and random effects. We then assessed the unconditional effect by regressing endline scores (post-intervention Coding and CT scores) against the treatment Condition (Treatment vs. Control), incorporating baseline scores as a covariate to account for initial performance variations. Finally, we developed a conditional model that included additional covariates to mitigate potential confounding factors and test differences between learner groups. All analyses were conducted in R (RStudio version 2023.03.0 + 386), using the lmtest (version 0.9–40; Hothorn et al., 2022) and clubSandwich (version 3.1-0; Pustejovsky, 2022) packages. Figure 5 details the models and variables used.

To evaluate model fit, we used a likelihood ratio test comparing our model to the intercept-only model (Grimm et al., 2016). Fixed effects were assessed with standard normal distribution *p*-values, and random effects' significance was determined through likelihood ratio tests (see Appendix). Given the non-normal distribution of baseline coding scores, a log transformation was applied. Robust standard errors were obtained using the Huber/White estimator (Maas & Hox, 2004). To ensure unbiased estimates, we employed bias-reduced linearization for calculating standard errors, confidence intervals, and *p*-values (Pustejovsky & Tipton, 2018). Given that coding and CT scores were on different scales, we normalized them to a 0-100 scale for easier comparison (Jain et al., 2005).

To address the study's second research question, a path analysis was conducted using bootstrapping (100 iterations) after confirming that all model assumptions were satisfied (Kline, 2016). The participant-to-variable ratio was approximately 87:1, with a sample size of 1,049 and 12 variables included in the analysis, suggesting that the sample was adequate to yield reliable estimates (Kline,

Variable Name	Description
$Student_n$	Unique identifier for each student.
$School_n$	Identifier for the school each student belongs to.
Site	State A (reference group) or State B.
Condition	Indicator variable for whether the school was allocated to receive the Treatment or Control (reference group).
Grade	Indicator variable for grade with three levels: K; Grade 1 (reference group) and Grade 2.
Gender	Indicator variable for student gender as identified by the school district with 2 levels: AFAB vs. AMAB (reference group).
Race/Ethnicity	Variable indicating the race/ethnicity of the student as reported by school districts at baseline. These included: White (reference group), Hispanic, Asian, Black or African American and Other.
IEP	Indicator variable for whether the student has an Individualized Education Program (IEP) = 1, as reported by school districts at baseline.
ELL	Indicator variable for whether the student is classified as English Language Learner (ELL) = 1, as reported by school districts at baseline.
SES	Indicator variable for whether the student comes from a low socioeconomic status (SES) background (measured by free or reduced lunch status = 1) as reported by school districts at baseline.
Dosage	Variable, centered around the grand mean, representing the amount of CS instructional time, measured by the last lesson taught from the CAL curriculum as reported by teachers. Each lesson lasts approximately 45 minutes, so the lesson number is multiplied by 45 to provide a rough estimation of instructional time. For example, if a teacher taught 14 lessons, the dosage is roughly estimated as 14×45=63014×45=630 minutes of CAL. Since students in the Control condition didn't receive the treatment, dosage is set to 1 minute, so this variable primarily reflects the effect of increased dosage within the Treatment condition.
Coding	Scores for the validated Coding Stages Assessment (CSA) measuring coding skills. This is an outcome variable and is measured before and after the curriculum introduction. The score is normalized to a scale of 0-100 and then log-transformed. The baseline score is group mean centered.
CT	Scores for the validated TechCheck assessment measuring computational thinking. This is an outcome variable and is measured before and after the curriculum introduction. The score is normalized to a scale of 0-100. The baseline score is group mean centered.
Model Name	**Description**
Null	This is an intercept-only model, containing only the intercept and random effects.
Unconditional	This model regresses the endline scores on the Condition, with baseline scores included as a covariate.
Conditional	This model regresses the endline scores on the Condition, with all predictors (including baseline scores) included as covariates.

Fig. 5 A summary of the MLM models fitted to explore predictors' impact on coding and CT skills growth, detailing the included variables

2016). Model fit was evaluated using the Chi-square goodness-of-fit test and additional indices (Hooper et al., 2008), including the root mean square error of approximation (RMSEA), comparative fit index (CFI), Tucker–Lewis index (TLI), and standardized root mean square residual (SRMR).

Results

Table 2 presents the means and standard deviations of baseline and endline scores for each outcome measure, broken down by treatment condition. The data indicate that students

Table 2 Descriptive Statistics for Students Coding and CT Scores by Condition (n = 1,705)

	Control				Treatment			
	Baseline		Endline		Baseline		Endline	
Outcome	*M*	*SD*	*M*	*SD*	*M*	*SD*	*M*	*SD*
Coding	9.72	7.49	16.89	12.82	9.73	7.95	31.49	17.36
CT	37.15	15.71	44.21	16.85	31.85	21.58	38.49	24.58

in both conditions achieved higher scores at endline compared to baseline. The conditional models for both Coding skills and CT skills demonstrated significantly improved fit over the null models (Coding: $\chi^2(15) = 817.27$, $p < .001$; CT: $\chi^2(15) = 1{,}264.21$, $p < .001$). The MLM results for both unconditional and conditional models are detailed in Table 3.

The conditional model showed higher marginal R^2 values for both Coding skills (0.456 vs. 0.348) and CT skills (0.546 vs. 0.388), indicating a greater proportion of variance explained by the fixed effects. It also had higher conditional R^2 values than the unconditional model (both fixed and random effects), with values of 0.474 vs. 0.430 for Coding skills and 0.658 vs. 0.643 for CT skills, demonstrating enhanced explanatory power. The reduction in standard deviations from the unconditional to the conditional model indicates that the covariates in the conditional model effectively capture variability between and within schools. This enhances the model's ability to explain outcomes, providing more accurate estimations of treatment effects and covariates on student performance. This improvement is further supported by the AIC and BIC values.

The ICC values for both Coding and CT skills decreased from the unconditional to the conditional model. For Coding, the ICC dropped from 0.1 (10%) to 0.0, indicating that the conditional model accounted for systematic differences between schools that affect coding scores. For CT, the ICC decreased from 0.4 (40%) to 0.2 (20%), with the remaining 80% of variance due to within-school differences. This shows that the added covariates in the conditional model effectively explain some of the between-school variance in CT scores, reducing the proportion of variance attributable to differences between schools.

In practical terms, this means that students who received the CAL curriculum showed significantly greater improvement in their coding and CT skills than their peers in control classrooms, even after accounting for differences in prior ability, grade level, and school context. While we present the results from both the unconditional and conditional models, our discussion will primarily focus on the conditional model, as it demonstrated superior fit. Due to space limitations, we highlight the results related to the overall effectiveness of the intervention. A more detailed discussion of other findings, including the intervention's impact on reducing disparities, can be found in the appendix.

The results of the path analysis model are presented in Table 4, while a visualization of the model is shown in Fig. 6. Model fit indices indicated a strong fit between the hypothesized model and the data: TLI = 0.97, CFI = 0.99, RMSEA = 0.05 (90% CI [0.02, 0.08]), and SRMR = 0.02. All values met or exceeded commonly accepted thresholds for good model fit (Hooper et al., 2008). The Chi-square goodness of fit test was statistically significant, $\chi^2(3) = 10.73$, $p = .013$. However, given the large sample size and strong performance on other fit indices, this result may reflect the sensitivity of the Chi-square test to sample size rather than a meaningful misfit (Hooper et al., 2008).

This suggests a clear pathway: students who showed greater growth in their coding skills also made greater gains in both math and literacy. However, growth in the broader skill of CT did not show a similar direct link to academic improvement in this analysis.

Discussion

RQ1. Effect of the Intervention

The intervention significantly improved students' coding and CT skills, as evidenced by the positive and significant coefficients for the treatment condition in the conditional multilevel model (Table 3). Students with stronger initial coding and CT skills demonstrated greater gains over time, a pattern consistent with educational interventions where prior knowledge facilitates new learning. Developmental differences were evident across grade levels: kindergarteners scored significantly lower than first and second graders, supporting the need for explicit, scaffolded instruction tailored to foundational skill development fully (Atmatzidou & Demetriadis, 2016; Cachero et al., 2020). This aligns with cognitive developmental theory, where working memory and abstract reasoning capacities expand during early childhood (Piaget et al., 1957).

Instructional dosage, measured as minutes of CAL instruction, played a distinct role for each skill set. More instructional time was associated with modest but significant gains in coding skills, reinforcing the value of sustained practice with programming tools like ScratchJr (Tondeur et al., 2019). In contrast, increased dosage did not correspond to similar gains in CT skills; in fact, the coefficient

Table 3 Multilevel regression model results for CS intervention impacts on student coding and CT skills ($n = 1{,}705$)

Parameter	Unconditional		Conditional	
	Coding	CT	Coding	CT
(Intercept)	2.692***	52.669***	2.981***	57.214***
	(0.074)	(2.615)	(0.080)	(2.272)
Condition (Treatment)	0.667***	−8.457***	0.232**	8.503*
	(0.086)	(2.043)	(0.073)	(3.564)
Baseline Coding skills (group-centered)	0.350***		0.249***	
	(0.025)		(0.023)	
Baseline CT skills (group-centered)		0.726***		0.716***
		(0.042)		(0.046)
Site (State B)			−0.222***	−16.834***
			(0.054)	(2.692)
Dosage (Minutes of CS; mean-centered)			0.001***	−0.009***
			(0.000)	(0.002)
Year (Year 2)			0.020	−9.766**
			(0.049)	(3.489)
Gender (AFAB)			−0.038	−0.849
			(0.024)	(0.659)
Grade (Grade 2)			0.099*	−0.858
			(0.046)	(0.914)
Grade (Kindergarten)			−0.258***	−3.521*
			(0.053)	(1.495)
Race/ethnicity (Asian)			0.128*	0.228
			(0.053)	(1.334)
Race/ethnicity (Black or African American)			−0.095	−1.950
			(0.062)	(1.665)
Race/ethnicity (Hispanic)			−0.124**	−1.860*
			(0.039)	(0.879)
Race/ethnicity (Mixed or Other)			−0.031	−0.779
			(0.066)	(1.185)
IEP (Yes)			−0.145***	−3.061**
			(0.044)	(1.082)
ELL (Yes)			−0.091**	0.620
			(0.030)	(1.564)
SES (Low)			−0.062*	1.407
			(0.027)	(0.879)
SD (Intercept School)	0.187	12.342	0.088	7.102
SD (Observations)	0.493	13.857	0.472	13.644
Number of Observations	1705	1679	1617	1592
R^2 Marginal	0.348	0.388	0.456	0.546
R^2 Conditional	0.430	0.658	0.474	0.643
AIC	2505.7	13705.1	2296.1	12913.6
BIC	2532.9	13732.2	2393.1	13010.3
ICC	0.1	0.4	0.0	0.2
RMSE	0.49	13.73	0.47	13.47

+ $p < .1$, * $p < .05$, ** $p < .01$, *** $p < .001$; Robust standard errors within parentheses

was negative, suggesting potential diminishing returns or that CT development may depend more on pedagogical quality than sheer instructional time. Previous research has suggested that AFAB students may require additional training time to achieve the same skill level as their counterpart (Atmatzidou & Demetriadis, 2016).

Implementation context also influenced outcomes. The intervention's effect on CT skills was significantly greater in Phase I (expert-led PD) than in Phase II (peer-led PD), as shown by the negative coefficient for Year 2 in the CT model (Table 3). This difference may reflect variation in the depth of PD support for teaching nuanced CT concepts. No such phase difference was observed for coding skills. Exploratory analyses of state-level differences, including baseline scores and instructional time, did not yield significant interaction effects with the treatment condition (e.g., State × Dosage). While these factors represent important real-world variation, they did not moderate the core finding

Table 4 Unstandardized loadings (Standard Errors), standardized loadings, and significance levels for each parameter in the path analysis model ($N = 1049$)

Parameter Estimate	Unstandardized	Standardized	*p*
Regressions			
Coding Growth→ Math Growth	0.22(0.07)	0.09	0.001
Coding Growth→ Literacy Growth	0.18(0.06)	0.09	0.001
Condition → Coding Growth	14.95(0.67)	0.53	<0.001
CT Growth→ Math Growth	0.18(0.06)	0.09	0.003
CT Growth→ Literacy Growth	0.13(0.06)	0.08	0.025
Condition → CT Growth	-0.12(1.06)	-0.004	0.907
Covariances			
Covariance for Math Growth and Literacy Growth	581.81(35.93)	0.56	<0.001
Errors			
Error in Coding Growth	146.38(9.38)	0.72	<0.001
Error in CT Growth	287.19(10.50)	1.00	<0.001
Error in Condition	0.25(0.0006)	1.00	<0.001
Error in Literacy Growth	867.67(37.87)	0.99	<0.001
Error in Math Growth	1,228.68(54.29)	0.98	<0.001

$\chi^2(3) = 10.73$, $p = .013$; Growth = (Endline score – Baseline score); Indirect Effects (Standardized): Condition → Coding Growth → Math/Literacy Growth: $0.53 \times 0.09 = 0.0477$; Condition → CT Growth → Math Growth: (-0.004) × 0.09 = -0.00036; Condition → CT Growth → Literacy Growth: (-0.004) × 0.08 = -0.00032

that the CAL intervention significantly improved coding and CT skills, indicating that the core intervention effect was robust. Detailed state-level results are available in the appendix.

RQ2. CS and Academic Development in Math and Literacy

The path analysis (Table 4; Fig. 6) provides clear evidence for the transfer of coding skills to core academic domains. Growth in coding skills significantly predicted gains in both math and literacy achievement, indicating a meaningful pathway through which coding instruction supports broader academic learning. In contrast, growth in CT skills did not show a significant direct relationship with academic gains in the path model, nor was there a significant direct effect of the treatment condition on CT growth within this analytic framework.

This divergence between coding and CT highlights an important distinction. The tangible, practiced skill of coding, involving the direct application of sequencing, logic, and debugging within ScratchJr, appears to foster cognitive and academic competencies that transfer directly to math and literacy tasks in the short term. The broader, more abstract construct of CT, while improved by the intervention as shown in the multilevel model, did not demonstrate a similar short-term transfer effect within our measurement model.

Several factors may explain this pattern. Coding assessments (e.g., the CSA) measure concrete, observable

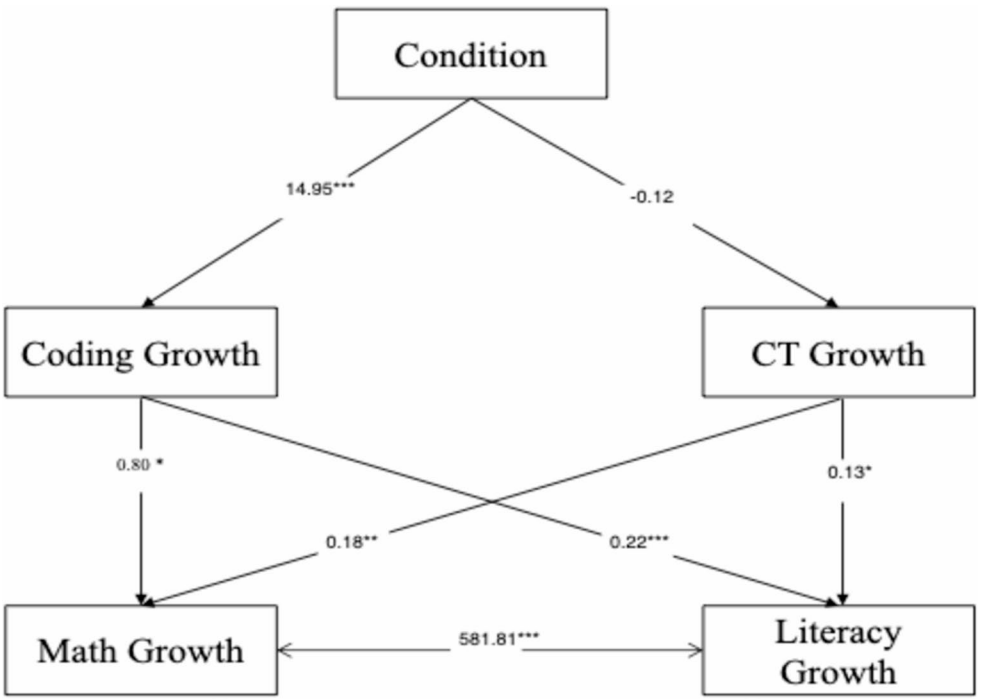

Fig. 6 A visualization of the path analysis model and results. *Note*: Growth = (Endline score – Baseline score). Indirect path and error terms are not shown to prevent overcrowding. Refer to Table 4 for the complete results

performance, whereas assessing the multifaceted, latent construct of CT in young learners presents well-documented methodological challenges (Lai & Ellefson, 2023), potentially attenuating detectable links to academic outcomes. Furthermore, the immediate academic benefits may be more directly tied to the applied, procedural practice of coding, while the influence of higher-order CT may be more distal and thus less captured within the study's timeframe.

The significant indirect effects observed through coding skills underscore the instructional value of integrating structured coding activities into early childhood education (Bers, 2019). Using ScratchJr, students engage with mathematical concepts such as counting, patterning, and spatial reasoning through parameters and loops, while creating animated stories fosters narrative sequencing, vocabulary, and expressive language skills. Beyond domain-specific knowledge, the iterative cycle of designing, testing, and debugging programs cultivates executive functions, including planning, working memory, and problem-solving, that are foundational to academic success (Chu et al., 2016; Voogt et al., 2015).

These findings suggest that early coding instruction offers more than technical skill development; it creates a conduit for strengthening transferable cognitive and academic competencies. While CT development remains a vital educational goal, its impact on academic achievement may unfold over a longer developmental trajectory or require more targeted pedagogical and assessment approaches to be fully realized. Future research should employ longitudinal designs and refine methods for assessing CT growth in young learners to better understand its contribution to academic development.

To contextualize these findings, it is important to note that participating schools were located in two U.S. East Coast settings: a large urban public school district and a mix of suburban and small-town districts. These contexts reflect typical variation in U.S. public education systems, including differences in access to instructional resources, availability of specialized staff, and scheduling flexibility for non-tested subjects such as computer science. In the urban district (State A), many schools serve historically under-resourced communities and operate under tighter constraints related to staffing, time allocation, and competing academic priorities. In contrast, districts in State B include a combination of relatively higher-resourced suburban schools and smaller districts, where resource allocation and instructional capacity can vary widely across schools.

Across both contexts, CS was generally not implemented as a standalone subject prior to the intervention, but rather integrated into other content areas without dedicated time or structured curricula. This variability in baseline conditions highlights that even within a single region, access to high-quality CS instruction is uneven. As such, the need for dedicated instructional time, sustained PD, and equitable resource allocation is not unique to this study, but reflects broader systemic patterns in U.S. elementary education, particularly in early-grade CS implementation.

Conclusion and Implications

The intervention improved students' coding and CT skills, underscoring the value of inclusive, age-appropriate CS curricula. The CAL curriculum, which combined unplugged and plugged activities, effectively engaged young learners in active, hands-on learning that promoted critical thinking and problem-solving from an early age. Findings from the second research question reveal a significant link between coding skill growth and gains in math and literacy, extending the value of early CS instruction beyond technical proficiency. These results carry broader implications for curriculum design, instructional strategies, and educational policy.

Implications for Curriculum Design

The success of the CAL intervention demonstrates that developmentally sequenced, literacy-integrated CS curricula can effectively build both coding proficiency and foundational CT skills in K-2 students. The observed transfer of coding skills to math and literacy achievement underscores the importance of designing curricula that intentionally connect computational practices (e.g., sequencing, debugging) to core academic concepts. Curriculum developers should prioritize scaffolded, play-based activities that blend unplugged and plugged learning, particularly in the early grades where foundational skill integration is critical.

Implications for Instructional Strategies

The positive association between instructional dosage and coding gains suggests that consistent, weekly coding practice supports skill development. However, the diminished returns for CT highlight that more time alone is insufficient; instructional quality and pedagogical focus matter. Educators should balance hands-on coding time with explicit discussion of CT concepts (e.g., decomposition, patterns) and leverage coding projects as contexts for academic skill practice, for example:

- Embed coding into literacy blocks by having students plan, sequence, and animate the events of a story using ScratchJr.

– Integrate coding into math lessons by creating programs that visually demonstrate counting, repeating patterns, or simple number stories.

The phase-related difference in CT outcomes further emphasizes the need for ongoing, high-quality professional development that deepens teachers' ability to facilitate CT-rich learning.

Implications for Educational Policy

This study provides empirical support for integrating CS into early elementary standards and schedules. The equitable improvements across diverse student groups, coupled with the academic transfer observed, strengthens the argument for universal, early access to CS education as a strategy for supporting broad academic development and reducing opportunity gaps. Policymakers and administrators should invest in teacher preparation and sustained PD, allocate dedicated instructional time for CS, and ensure resource equity so that all schools, especially those serving low-SES communities, can implement high-quality, inclusive CS curricula like CAL.

Limitations

Although the study aimed to assess the overall impact of CS education on K-2 students, we anticipated variability across learner groups. However, district participation requirements limited our ability to obtain a balanced sample. The sample reflects real-world conditions, with randomization occurring at the school level. Stratification by poverty level was only feasible in one state, reflecting common research challenges in school-based studies. These sampling constraints may affect the generalizability of findings to other regional or demographic contexts.

Also, the PD model shifted from expert-led in Year 1 to peer-led in Year 2, with facilitation provided by teachers who had already completed the initial training and implementation. While this approach aimed to promote sustainability and leveraged experienced practitioners, it may have introduced variability in the depth or consistency of training, particularly for the more abstract pedagogy required to teach computational thinking (CT) concepts. This real-world scaling decision is an important consideration when interpreting the results and highlights the critical role of sustained, high-quality support for teachers implementing complex new curricula. Consequently, the observed phase difference in CT outcomes should be interpreted in light of this implementation variation, not solely as a pure curriculum effect.

The COVID-19 pandemic required a shift to online assessment and instruction for part of the intervention, which was not originally planned. This may have influenced both the delivery fidelity of lessons and the nature of student engagement. Additionally, the agreement with the district did not include control group data collection during Phase II, limiting longitudinal analysis. This precludes a clean two-year comparison of treatment versus control and focuses the analysis on within-phase effects. Other limitations are discussed throughout the manuscript.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s10643-026-02252-2.

Acknowledgements We extend our gratitude to the students, teachers, and site coordinators for their contributions to this study. Special thanks to the Shaffer Evaluation Group for their supervision and the Scratch Foundation for their support. We also appreciate the valuable feedback from the reviewers during the review process.

Declarations

Competing Interests The authors declare no competing interests. This project was funded by the U.S. Department of Education. The curriculum underlying this research was developed by the second author and is freely available online.

References

Alrawashdeh, G. S., Nadler, E. C., & Bers, M. U. (2024). *Virtual professional development enhances elementary teacher' coding skills and self-efficacy: A comparison of three models*. In S. Papadakis & M. Kalogiannakis (Eds.), Advances in Early Childhood and K-12 Education (pp. 114–137). IGI Global. https://doi.org/10.4018/979-8-3693-2377-9.ch005

Alrawashdeh, G. S., Bergman, A. J., & Bers, M. U. (2025). From teacher training to student growth: Virtual professional d evelopment enhances K–2 computer science education introduction. *Early Childhood Education Journal*. https://doi.org/10.1007/s10643-025-01961-4

Angeli, C., & Giannakos, M. (2020). Computational thinking education: Issues and challenges. *Computers in Human Behavior*, *105*, 106185. https://doi.org/10.1016/j.chb.2019.106185

Atmatzidou, S., & Demetriadis, S. (2016). Advancing students' computational thinking skills through educational robotics: A study on age and gender relevant differences. *Robotics and Autonomous Systems*, *75*, 661–670. https://doi.org/10.1016/j.robot.2015.10.008

Barcelos, T. S., Muñoz-Soto, R., Villarroel, R., Merino, E., & Silveira, I. F. (2018). Mathematics Learning through Computational Thinking Activities: A Systematic Literature Review. *J Univers Comput Sci*, *24*, 815–845.

Bati, K. (2022). A systematic literature review regarding computational thinking and programming in early childhood education. *Education and Information Technologies*, *27*(2), 2059–2082. https://doi.org/10.1007/s10639-021-10700-2

Battista, M. T., & Clements, D. H. (1986). The effects of Logo and CAI problem-solving environments on problem-solving abilities and mathematics achievement. *Computers in Human Behavior*, *2*(3), 183–193. https://doi.org/10.1016/0747-5632(86)90002-6

Bers, M. U. (2017). *Coding as a playground: Programming and computational thinking in the early childhood classroom* (1st edition). 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., Strawhacker, A., & Sullivan, A. (2022). *The state of the field of computational thinking in early childhood education* (OECD Education Working Papers No. 274; OECD Education Working Papers, Vol. 274). https://doi.org/10.1787/3354387a-en

Bloom, H. S., Richburg-Hayes, L., & Black, A. R. (2007). Using covariates to improve precision for studies that randomize schools to evaluate educational interventions. *Educational Evaluation and Policy Analysis*, *29*(1), 30–59. https://doi.org/10.3102/0162373707299550

Cachero, C., Barra, P., Meliá, S., & López Granado, O. (2020). Impact of programming exposure on the development of computational thinking capabilities: An empirical study. *IEEE Access*, *8*, 72316–72325. https://doi.org/10.1109/ACCESS.2020.2987254

Century, J., Ferris, K. A., & Zuo, H. (2020). Finding time for computer science in the elementary school day: A quasi-experimental study of a transdisciplinary problem-based learning approach. *International Journal of STEM Education*, *7*(1), 20. https://doi.org/10.1186/s40594-020-00218-3

Cheng, L., Wang, X., & Ritzhaupt, A. D. (2023). The effects of computational thinking integration in STEM on students' learning performance in K-12 education: A meta-analysis. *Journal of Educational Computing Research*, *61*(2), 416–443. https://doi.org/10.1177/07356331221114183

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 & Education*, *109*, 162–175. https://doi.org/10.1016/j.compedu.2017.03.001

Chu, F. W., vanMarle, K., & Geary, D. C. (2016). Predicting children's reading and mathematics achievement from early quantitative knowledge and domain-general cognitive abilities. *Frontiers in Psychology*, *7*. https://doi.org/10.3389/fpsyg.2016.00775

Claessens, A., & Engel, M. (2013). How important is where you start? Early mathematics knowledge and later school success. *Teachers College Record*, *115*(6), 1–29. https://doi.org/10.1177/016146811311500603

de Ruiter, L. E., & Bers, M. U. (2022). The Coding Stages Assessment: Development and validation of an instrument for assessing young children's proficiency in the ScratchJr programming language. *Computer Science Education*, *32*(4), 388–417. https://doi.org/10.1080/08993408.2021.1956216

Dohn, N. B. (2020). Students' interest in Scratch coding in lower secondary mathematics. *British Journal of Educational Technology*, *51*(1), 71–83. https://doi.org/10.1111/bjet.12759

Doleck, T., Bazelais, P., Lemay, D. J., Saxena, A., & Basnet, R. B. (2017). Algorithmic thinking, cooperativity, creativity, critical thinking, and problem solving: Exploring the relationship between computational thinking skills and academic performance. *Journal of Computers in Education*, *4*(4), 355–369. https://doi.org/10.1007/s40692-017-0090-9

Durak, H. Y., & Saritepeci, M. (2018). Analysis of the relation between computational thinking skills and various variables with the structural equation model. *Computers & Education*, *116*, 191–202. https://doi.org/10.1016/j.compedu.2017.09.004

Egbert, J., Shahrokni, S. A., Abobaker, R., & Borysenko, N. (2021). It's a chance to make mistakes: Processes and outcomes of coding in 2nd grade classrooms. *Computers & Education*, *168*, 104173. https://doi.org/10.1016/j.compedu.2021.104173

Fagerlund, J., Häkkinen, P., Vesisenaho, M., & Viiri, J. (2021). Computational thinking in programming with Scratch in primary schools: A systematic review. *Computer Applications in Engineering Education*, *29*(1), 12–28. https://doi.org/10.1002/cae.22255

Friend, M., Matthews, M., Winter, V., Love, B., Moisset, D., & Goodwin, I. (2018). Bricklayer: Elementary students learn math through programming and art. *In Proceedings of the 49th ACM Technical Symposium on Computer Science Education (SIGCSE '18),* pp 628–633. https://doi.org/10.1145/3159450.3159515

Fuchs, L. S., Bucka, N., Clarke, B., Dougherty, B., Jordan, N. C., Karp, K. S., Woodward, J., Jayanthi, M., Gersten, R., Newman-Gonchar, R., Schumacher, R., Haymond, K., Lyskawa, J., Keating, B., Morgan, S., & Jacobson, J. (2021). *Assisting students struggling with mathematics: Intervention in the elementary grades*. Institute of Education Sciences. https://whatworks.ed.gov/

Gülmez, I., & Özdener, N. (2015). Academic achievement in computer programming instruction and effects of the use of visualization tools; at the elementary school level. *Journal of Education Society and Behavioural Science*, *11*(1), 1–18. https://doi.org/10.9734/BJESBS/2015/18316

Goldstein, H. (2011). *Multilevel statistical models* (4th ed.). John Wiley & Sons.

Grimm, K. J., Ram, N., & Estabrook, R. (2016). *Growth modeling: Structural equation and multilevel modeling approaches*. Guilford Press.

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

Haseski, H., Ilic, U., & Tugtekin, U. (2018). Defining a new 21st century skill-computational thinking: Concepts and trends. *International Education Studies*, *11*(4), 4. https://doi.org/10.5539/ies.v11n4p29

Hassenfeld, Z. R., Govind, M., Ruiter, L. E. D., & Bers, M. U. (2020). If you can program, you can write: Learning introductory programming across literacy levels. *Journal of Information Technology Education: Research*, *19*, 065–085.

Hickmott, D., Prieto-Rodriguez, E., & Holmes, K. (2018). A scoping review of studies on computational thinking in K–12 mathematics classrooms. *Digital Experiences in Mathematics Education*, *4*(1), 48–69. https://doi.org/10.1007/s40751-017-0038-8

Hooper, D., Coughlan, J., & Mullen, M. (2008). Structural equation modelling: Guidelines for determining model fit. *Electronic Journal of Business Research Methods*, *6*(1), 53–60. https://doi.org/10.21427/D7CF7R

Hothorn, T., Bretz, F., & Westfall, P. (2022). multcomp [Computer software]. https://CRAN.R-project.org/package=multcomp

Hsu, T. C., Chang, S. C., & Hung, Y. T. (2018). How to learn and how to teach computational thinking: Suggestions based on a review of the literature. *Computers & Education*, *126*, 296–310. https://doi.org/10.1016/j.compedu.2018.07.004

Hu, L., & Wang, H. (2024). Unplugged activities in the elementary school mathematics classroom: The effects on students' computational thinking and mathematical creativity. *Thinking Skills and Creativity*, *54*, 101653. https://doi.org/10.1016/j.tsc.2024.101653

Jain, A., Nandakumar, K., & Ross, A. (2005). Score normalization in multimodal biometric systems. *Pattern Recognition*, *38*(12), 2270–2285. https://doi.org/10.1016/j.patcog.2005.01.012

Kafai, Y., & Burke, Q. (2014). *Connected code: Why children need to learn programming*. The MIT Press.

Kastberg, D., Chan, J. Y., & Murray, G. (2015). *Performance of U.S. 15-year-old students in science, reading, and mathematics literacy in an international context-first look at PISA 2015*. National Center for Education Statistics. https://nces.ed.gov/pubs2017/2017048.pdf

Kaup, C. F., Pedersen, P. L., & Tvedebrink, T. (2023). Integrating computational thinking to enhance students' mathematical understanding. *Journal of Pedagogical Research*, *7*(2), 127–142.

Kite, V., Park, S., & Wiebe, E. (2021). The code-centric nature of computational thinking education: A review of trends and issues in computational thinking education research. *Sage Open*, *11*(2), 21582440211016418. https://doi.org/10.1177/21582440211016418

Kline, R. B. (2016). *Principles and practice of structural equation modeling* (4th ed.). Guilford Press.

Korkmaz, Ö. (2012). The impact of critical thinking and logico-mathematical intelligence on algorithmic design skills. *Journal of Educational Computing Research*, *46*(2), 173–193. https://doi.org/10.2190/EC.46.2.d

Lai, R. P. Y., & Ellefson, M. R. (2023). How multidimensional is computational thinking competency? A bi-factor model of the computational thinking challenge. *Journal of Educational Computing Research, 61*(2), 259–282. https://doi.org/10.1177/07356331221121052

Lei, H., Chiu, M. M., Li, F., Wang, X., & Geng, Y. (2020). Computational thinking and academic achievement: A meta-analysis among students. *Children and Youth Services Review*, *118*, 105439. https://doi.org/10.1016/j.childyouth.2020.105439

Lewis, D. W., Kohne, L., Mechlinski, T., & Schmalstig, M. (2015). The exploring computer science course, attendance and math achievement. *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education, ITiCSE '15*, 147–152. https://doi.org/10.1145/2729094.2742598

Limiñana Gras, R. M., Bordoy, M., Ballesta, G. J., & Berna, J. C. (2010). Creativity, intellectual abilities and response styles: Implications for academic performance in the secondary school. *Anales de Psicología*, *26*(2), 212–219.

Li, S., Liu, X., Yang, Y., & Tripp, J. (2022). Effects of teacher professional development and science classroom learning environment on teachers' science achievement. *Research in Science Education*, *52*, 1031–1053. https://doi.org/10.1007/s11165-020-09979-x

Maas, C. J. M., & Hox, J. J. (2004). Robustness issues in multilevel regression analysis. *Statistica Neerlandica*, *58*(2), 127–137. https://doi.org/10.1046/j.0039-0402.2003.00252.x

Messer, D., Thomas, L., Holliman, A., & Kucirkova, N. (2018). Evaluating the effectiveness of an educational programming intervention on children's mathematics skills, spatial awareness and working memory. *Education and Information Technologies*, *23*(6), 2879–2888. https://doi.org/10.1007/s10639-018-9747-x

Miller, J. (2019). STEM education in the primary years to support mathematical thinking: Using coding to identify mathematical structures and patterns. *Zdm Mathematics Education*, *51*(6), 915–927. https://doi.org/10.1007/s11858-019-01096-y

Moreno-León, J., Román-González, M., & Robles, G. (2018). On computational thinking as a universal skill: A review of the latest research on this ability. *2018 IEEE Global Engineering Education Conference (EDUCON)*, 1684–1689. https://doi.org/10.1109/EDUCON.2018.8363437

Papadakis, S. (2024). Can preschoolers learn computational thinking and coding skills with ScratchJr? A systematic literature review. *International Journal of Educational Reform*, *33*(1), 28–61. https://doi.org/10.1177/10567879221076077

Papert, S. (1993). *Mindstorms: Children, computers, and powerful ideas* (2nd ed.). Basic Books.

Piaget, J. (1973). *The child and reality: Problems of genetic psychology* (A. Rosin, Trans.). Grossman.

Piaget, J., Inhelder, B., Langdon, F. J., & Lunzer, J. L. (1957). The Child's Conception of Space. *British Journal of Educational Studies*, *5*(2), 187–189. https://doi.org/10.2307/3118882

Psycharis, S., & Kallia, M. (2017). The effects of computer programming on high school students' reasoning skills and mathematical self-efficacy and problem solving. *Instructional Science*, *45*(5), 583–602. https://doi.org/10.1007/s11251-017-9421-5

Pustejovsky, J. E. (2022). *clubSandwich: Cluster-Robust (Sandwich) Variance Estimators with Small-Sample Corrections* (Version 0.5.8) [Computer software]. https://CRAN.R-project.org/package=clubSandwich

Pustejovsky, J. E., & Tipton, E. (2018). Small-sample methods for cluster-robust variance estimation and hypothesis testing in fixed effects models. *Journal of Business & Economic Statistics*, *36*(4), 672–683. https://doi.org/10.1080/07350015.2016.1247004

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*, *29*(4), 482–498. https://doi.org/10.1007/s10956-020-09831-x

Salac, J., Thomas, C., Butler, C., & Franklin, D. (2021). Understanding the link between computer science instruction and reading & math performance. *Proceedings of the 26th ACM Conference on Innovation and Technology in Computer Science Education V 1 ITiCSE '21*, *408*, 414. https://doi.org/10.1145/3430665.3456313

Salac, J., Thomas, C., Twarek, B., Marsland, W., & Franklin, D. (2020). Comprehending code: Understanding the relationship between reading and math proficiency, and 4th-grade CS learning outcomes. *Proceedings of the 51st ACM Technical Symposium on Computer Science Education, SIGCSE '20*, 268–274. https://doi.org/10.1145/3328778.3366822

Selby, C., Woollard, J., & Woollard, J. (2014). Computational thinking: The developing definition. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education*.

Seoane Pardo, A. M. (2018). Computational thinking between philosophy and STEM—programming decision making applied to the behavior of moral machines in ethical values classroom. *IEEE Revista Iberoamericana de Tecnologias Del Aprendizaje*, *13*(1), 20–29. https://doi.org/10.1109/RITA.2018.2809940

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

Sun, L., Hu, L., & Zhou, D. (2022). Programming attitudes predict computational thinking: Analysis of differences in gender and programming experience. *Computers & Education*, *181*, 104457. https://doi.org/10.1016/j.compedu.2022.104457

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

Thompson, J., & Childers, G. (2021). The impact of learning to code on elementary students' writing skills. *Computers & Education*, *175*, 104336. https://doi.org/10.1016/j.compedu.2021.104336

Tikva, C., & Tambouris, E. (2021). Mapping computational thinking through programming in K-12 education: A conceptual model based on a systematic literature Review. *Computers & Education*, *162*, 104083. https://doi.org/10.1016/j.compedu.2020.104083

Tondeur, J., Scherer, R., Baran, E., Siddiq, F., Valtonen, T., & Sointu, E. (2019). Teacher educators as gatekeepers: Preparing the next generation of teachers for technology integration in education. *British Journal of Educational Technology*, *50*(3), 1189–1209. https://doi.org/10.1111/bjet.12748

Upadhyaya, B., McGill, M. M., & Decker, A. (2020). A longitudinal analysis of K-12 computing education research in the United States: Implications and recommendations for change. *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, *605*, 611. https://doi.org/10.1145/3328778.3366809

Vee, A. (2017). *Coding literacy: How computer programming is changing writing*. The MIT Press. https://direct.mit.edu/books/

monograph/3543/Coding-LiteracyHow-Computer-Programming-Is

Voogt, J., Fisser, P., Good, J., Mishra, P., & Yadav, A. (2015). Computational thinking in compulsory education: Towards an agenda for research and practice. *Education and Information Technologies*, *20*(4), 715–728. https://doi.org/10.1007/s10639-015-9412-6

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

Whyte, R., Ainsworth, S., & Medwell, J. (2020). Designing multimodal composition activities for integrated K-5 programming and storytelling. In M. Gresalfi & I. S. Horn (Eds.), *The interdisciplinarity of the learning sciences: 14th International Conference of the Learning Sciences (ICLS) 2020, Volume 3* (pp. 1317–1324). International Society of the Learning Sciences. https://doi.org/10.22318/icls2020.1317

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

Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society A: Mathematical Physical and Engineering Sciences*, *366*(1881), 3717–3725. https://doi.org/10.1098/rsta.2008.0118

Wing, J. M. (2019). *A conversation about computational thinking* (Occasional paper series). Education: Future Frontiers. https://education.nsw.gov.au/content/dam/main-education/teaching-and-learning/education-for-a-changing-world/media/documents/Computational-Conversation_1_A.pdf

Yadav, A., Zhou, N., Mayfield, C., Hambrusch, S., & Korb, T. J. (2011). Introducing computational thinking in education courses. *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education*, *465*, 470. https://doi.org/10.1145/1953163.1953297

Yang, T. C., & Lin, Z. S. (2024). Enhancing elementary school students' computational thinking and programming learning with graphic organizers. *Computers & Education*, *209*, 104962. https://doi.org/10.1016/j.compedu.2023.104962

Ye, H., Liang, B., Ng, O. L., & Chai, C. S. (2023). Integration of computational thinking in K-12 mathematics education: A systematic review on CT-based mathematics instruction and student learning. *International Journal of STEM Education*, *10*(1), 3. https://doi.org/10.1186/s40594-023-00396-w

Özcan, M. Ş., Çetinkaya, E., Göksun, T., & Kisbu-Sakarya, Y. (2021). Does learning to code influence cognitive skills of elementary school children? Findings from a randomized experiment. *British Journal of Educational Psychology*, *91*(4), e12429. https://doi.org/10.1111/bjep.12429

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