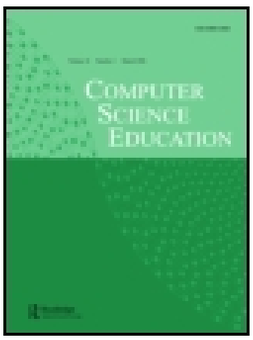

Computer Science Education

ISSN: (Print) (Online) Journal homepage: https://www.tandfonline.com/loi/ncse20

Teaching computational thinking to exceptional learners: lessons from two inclusive classrooms

Yenda Prado, Sharin Jacob & Mark Warschauer

To cite this article: Yenda Prado, Sharin Jacob & Mark Warschauer (2021): Teaching computational thinking to exceptional learners: lessons from two inclusive classrooms, Computer Science Education, DOI: 10.1080/08993408.2021.1914459

To link to this article: https://doi.org/10.1080/08993408.2021.1914459

Published online: 23 Apr 2021.

COMPUTER SCIENCE EDUCATION
https://doi.org/10.1080/08993408.2021.1914459

Routledge
Taylor & Francis Group

ARTICLE

Teaching computational thinking to exceptional learners: lessons from two inclusive classrooms

Yenda Prado, Sharin Jacob and Mark Warschauer

School of Education, University of California, Irvine, California, USA

ABSTRACT

Background and Context: Computational Thinking (CT) is a skill all students should learn. This requires using inclusive approaches to teach CT to a wide spectrum of students. However, strategies for teaching CT to students with exceptionalities are not well studied.
Objective: This study draws on lessons learned in two fourth-grade classrooms – one an inclusive general education classroom including students with and without disabilities, the other an inclusive GATE classroom including students with and without giftedness – to illustrate how CT frameworks can inform inclusive CS instruction.
Method: A comparative case study design integrating content analysis and first and second cycle coding of data was used to analyze teachers' instructional strategies using a CT framework. Data included transcriptions of audio-recorded classroom lessons, field notes, and conversations with teachers and students.
Findings: While each teacher used different strategies, both were effective in developing students' CT. Explicit instruction provided students receiving special education services with needed structure for the complex tasks inherent to computing. Peer feedback facilitated independent computational practice opportunities for students receiving GATE.
Implications: This study highlights how inclusive instructional practices can be assessed using a CT framework and leveraged to maximize learning and access to CT curricula for learners with exceptionalities.

ARTICLE HISTORY
Received 6 July 2020
Accepted 31 March 2021

KEYWORDS
CT; learners with exceptionalities; special education; gifted education; inclusive instruction; elementary education

Introduction

Our increasingly digitized world requires citizens with fundamental computational literacies to meet the professional demands of the economy and fulfill their public roles in society (Gover & Pea, 2018; Grover & Pea, 2013; Smith, 2016; Tissenbaum et al., 2019; Wing, 2006). As such, computational skills are increasingly relevant to innovating within diverse professions in industries from engineering to medicine and humanities to business (Grover & Pea, 2013; Vogel et al., 2017). In preparing students to the needs of this changing professional landscape (Gover & Pea, 2018; Martin et al., 2015), initiatives such as Computer Science (CS) for All aim to develop computational literacies in students from diverse backgrounds (Santo et al., 2019; Vogel et al., 2017). Beyond laying the foundation to increase representation in the supply of talented students graduating with CS degrees

CONTACT Yenda Prado ygiovana@uci.edu School of Education, University of California, Irvine, CA, USA

(Gover & Pea, 2018; Grover & Pea, 2013), CS education aims to develop students' computational agency, cultivate creativity capacities, and produce computationally literate citizens who can harness the potential of computing to address problems at work and in society (Mishra & Yadav, 2013; Santo et al., 2019; Smith, 2016; Tissenbaum et al., 2019; Vogel et al., 2017).

As broadening CS education access continues to grow in importance, we will be hard-pressed to learn more about, and move toward, developing strategies for equitably teaching CT to learners across diverse contexts (Gover & Pea, 2018; Grover & Pea, 2013; Santo et al., 2019). This requires approaches that address inherent challenges in teaching CT: bridging the digital divide faced by students with differing levels of computational proficiency and technological access who are entering a world requiring computational literacy (Martin et al., 2015; Mishra & Yadav, 2013; Smith, 2016; Vogel et al., 2017). While research on methods for broadening participation in CS education for under-represented students is growing (Goode et al., 2018; Grover & Pea, 2013; Ladner & Israel, 2016; Santo et al., 2019), there is less research exploring precisely how teacher instruction can be leveraged to help students with exceptionalities successfully access CS curricula.

Within the field of CS education, computational thinking (CT) is identified as a critical skill needed to address 21st century problems (Gover & Pea, 2018; Grover & Pea, 2013; Smith, 2016; Wing, 2006). As a result, there has been a great push to integrate CT into K-12 education (Goode et al., 2018; Grover & Pea, 2013). Broadly defined, CT refers to processes used to formulate thoughts and questions in a manner interpretable by computers to achieve desired results (Wing, 2006). Originally situated as a discrete skill to be learned within the confines of CS classes, CT has emerged as an increasingly diversified repertoire of approaches to problem solving and creating that all K-12 students should learn (Mishra & Yadav, 2013; Smith, 2016; Tissenbaum et al., 2019) now being integrated into multiple K-12 subjects including math (Pérez, 2018), science (Basu et al., 2017), and literacy (Jacob & Warschauer, 2018).

This re-envisioning of CS education aims requires approaches that meet the needs of the diverse array of students in US public schools, an as of yet understudied topic (see Hansen et al., 2016; Israel et al., 2017; Santo et al., 2019). In this paper, we preface our discussion of a dominant CT framework (Brennan & Resnick, 2012) with an overview of literature to date in inclusive CS education and the teaching of CT to learners with exceptionalities. We then use Brennan and Resnick (2012) CT framework to provide a description and explanation of how such frameworks might be used to uncover and examine the moves teachers and learners make in engaging with CT concepts, practices, and perspectives. We draw on two classroom cases – one an inclusive classroom comprised of students with and without disabilities, the other an inclusive Gifted and Talented Education (GATE) classroom comprised of students with and without giftedness – to illustrate the application of Brennan and Resnick's framework to better understand the ways, these teachers taught CT across diverse classroom contexts to learners with exceptionalities.

Teaching CT to learners with exceptionalities

Classroom variability is inherent to the 21st century educational landscape. Multiple factors account for varied abilities, including students with exceptionalities, students learning English as a second language, and students from under-resourced communities. At the national level, there have been efforts (e.g., CSforALL) to develop computing materials and strategies aimed at reaching all learners (Margolis & Goode, 2016; Smith, 2016) was launched to broaden participation in computing for traditionally underrepresented students in the field. Acting toward this goal, efforts have been made to dispel stereotypes about who does CS (Margolis, 2010) and resist sorting biases that determine student potential. Inclusive pedagogies, for example, those incorporating Universal Design for Learning (UDL) principles and culturally relevant approaches, have been found to bolster inclusivity (Barron & Darling-Hammond, 2008; Hitchcock & Stahl, 2003). Inclusive pedagogies seek to contextualize instructional content to students' needs and lives (Barron & Darling-Hammond, 2008; National Center on Universal Design for Learning, 2009). Despite the tenets of inclusive CS education, teacher professional development has largely focused on content knowledge, with only recent movement toward developing and integrating inclusive teaching strategies – both necessary for effective instruction of diverse learners, including those with exceptionalities (Darling-Hammond, 2012).

Movement towards inclusive CS education

Inclusive CS education has begun to focus on making content accessible for students with diverse abilities and needs. Effective practices for these students include, for example, integrating principles from inclusive instructional frameworks, such as UDL, into CT instruction (Basham et al., 2010; Israel et al., 2017; Israel, Ribuffo, & Smith, 2014). Differentiated, personalized instruction lies at the heart of inclusive instructional principles, the benefits of which have been well established for diverse students (see Hitchcock & Stahl, 2003).

As such, there is a growing interest in understanding how inclusive instructional principles manifest in classrooms to meet the needs of students with exceptionalities (Israel, Wherfel et al., 2015), and how these principles can inform curricular adaptations for diverse learners (Hansen et al., 2016). However, little research focuses on the manifestation of these principles vis-à-vis the specific moves teachers and learners make together as they connect to CT curricula. Consequently, utilizing frameworks to better understand how teaching and learning CT is constructed are essential to understanding how classroom dynamics might be structured to leverage and build on the talents and resources of learners with varied abilities and needs (Dudley-Marling & Burns, 2014; Israel, Pearson et al., 2015; Snodgrass et al., 2016).

An inclusive framework for understanding CT

Central to the premise of inclusive instructional principles is the use of inclusive frameworks that build on the competencies of learners with multiple differences and abilities (Dudley-Marling & Burns, 2014; Israel, Pearson et al., 2015; Rose et al., 2005). Such frameworks center proactive planning for learner variability to make instruction more inclusive

by providing multiple means of representation, expression, and engagement that all learners can access to develop their understanding of computational concepts, practices, and perspectives (Brennan & Resnick, 2012; Rose et al., 2005). For instruction involving learners with exceptionalities, inclusive uses of technologies are often operationalized under the umbrella of universal design (Edyburn, 2013).

Universal design has paved the way for inclusive instructional practices that afford learners multiple ways to access, use, and express information, demonstrate knowledge and skills, and engage with content (Edyburn, 2013; Rose et al., 2005). We define inclusive instructional practices as those that address the needs of students with and without exceptionalities holding a variety of abilities and needs (Dudley-Marling & Burns, 2014). In this context, inclusive classrooms are those that, in implementing inclusive instructional practices, support an integrated environment in which all students' contributions are equitably supported and valued (Algozzine & Ysseldyke, 2006; Sailor & Roger, 2005; Causton-Theoharis & Theoharis, 2008; Hall et al., 2004; Obiakor et al., 2012). Non-inclusive classrooms are those that privilege-specific ranges of ability and need deemed normative through the exclusion or segregation of students whose abilities and needs falling outside the prescribed norm (Armstrong et al., 2011; Dudley-Marling & Burns, 2014; Obiakor et al., 2012). We argue that CT frameworks can support inclusive instructional practice, toward the development of inclusive classrooms, by making visible the effective moves that teachers and students make in the cultivation of their CT concepts, practices, and perspectives.

Applying CT frameworks in the inclusive classroom

Here, we present an abbreviated version of Brennan and Resnick (2012) framework for CT to 1) outline inclusive ways to promote learners with exceptionalities' agency as computational thinkers and 2) provide a means of understanding how inclusive classrooms might develop CT across three key areas: concepts, practices, and perspectives. We do this through a description of how Brennan and Resnick (2012) framework – exemplified with instructional moves one might see in inclusive classrooms – can be used to engage with CT concepts, practices, and perspectives (for a thorough explanation of the CT framework, see *Using artifact-based interviews to study the development of CT in interactive media design*, Brennan & Resnick, 2012).

Computational concepts in the inclusive classroom. Computational concepts are the concepts designers engage with as they program (Brennan & Resnick, 2012). Common to many programming languages, computational concepts include: sequences (series of individual steps), loops (repeat function), parallelism (sequences occurring simultaneously), events (one thing causing another), conditionals ("if . . . . then . . . " clauses), operators (support for expressions), and data (storing, updating, retrieving values) (for more detailed explanation, see Brennan & Resnick, 2012). We apply Brennan and Resnick's framework to our study of inclusive classroom practices (see below) by examining frequency and intensity of discussion around each of these specified conceptual terms. We look at how explication of these terms surface within the Scratch curriculum and focus our summary of computational concepts to those that are most common, in this case: sequences, events, and conditionals. Sequences refer to a specific task or event being expressed by a series of specific steps or instructions (i.e., brushing your teeth; see case

studies below). Brennan and Resnick (2012) compare sequences to recipes by specifying a set of behaviors needed to produce a specific result. Events refer to one thing causing another to happen, for example, the start button initiating the commencement of audio elements in a programmed piece. Conditionals are the conditions that programmers put in place to dictate when and where an action takes place and often use the form "If . . . . Then . . . ." Use of conditionals in this case supports multiple outcomes in computer programming (Brennan & Resnick, 2012). Algorithms, while not explicitly listed by Brennan & Resnick, is another frequent computational concept that repeatedly surfaced in the inclusive classrooms we studied (see Findings). In these instances, algorithms were sometimes used interchangeably with sequences to refer to a finite series of instructions used to solve a problem or perform a specific computational task.

Computational practices in the inclusive classroom. Computational practices focus on the processes designers use to "move beyond what you are learning to how you are learning" (Brennan & Resnick, 2012, p. 7). Computational practices are used to engage with computational concepts and are characterized by Brennan and Resnick (2012) across four domains: being incremental and iterative (adaptive development process), testing and debugging (strategies for dealing with problems), reusing and remixing (building on other people's work), and abstracting and modularizing (building something large by putting together collections of smaller parts). In our study of inclusive classroom practice, being incremental and iterative and testing and debugging were the most developed practices we observed teachers using to engage students in CT (see Findings). In this context, being incremental and iterative is expressed in whole-group planning processes used in inclusive classrooms in the development of projects. This process could be characterized by approaching computational roadblocks or problems using smaller scaffolded steps, for example, by brainstorming solutions, trying them out as a class, and then modifying code based on results (see Findings). Critical aspects of testing and debugging include identifying the source of the problem, reading and experimenting with coding, writing and revising coding scripts until they work, and engaging with fellow coders to solve problems. The ultimate goal of testing and debugging is the enabling of more complex projects than what might have otherwise been created (Brennan & Resnick, 2012). The computational practice of *testing and debugging* might manifest itself in an inclusive classroom's use of strategies for managing and anticipating problems with students' coding. This can be seen in the ways an inclusive classroom might encourage students to independently engage in trial and error, subsequently take lessons learned from prior activities, and seek support from fellow coders, to test and debug their coding (Brennan & Resnick, 2012).

Computational perspectives in the inclusive classroom. Computational perspectives are those that designers form about themselves and the world (Brennan & Resnick, 2012). In the context of the inclusive classroom, computational perspectives build from teachers and students' computational practices with the goal of "evolving understandings of themselves, their relationships to others, and the technological world around them" (Brennan & Resnick, 2012, p. 10). Brennan and Resnick (2012) conceptualize these shifts in computational perspectives as *expressing* (computation for self-expression), *connecting* (computing with others), and *questioning* (computation for sense-making), with *connecting* and *questioning* being the more salient perspectives in our case study classrooms (see Findings). *Questioning* as a computational perspective touches on the idea of using

technology to ask questions to develop a better understanding of the world (Brennan & Resnick, 2012). We might see this in the inclusive classroom in attempts to make real-world and interdisciplinary connections between CT curricula and students' lived experiences. In this way, *questioning* is used to develop what one of Brennan and Resnick's student interviewee's termed "the programmer's mind" – the goal being to use our developing understanding of computation to better understand the world around us. One way this surfaces is through whole-class interrogation of process and design.

Connecting as a computational perspective touches on the social practices involved in engaging in computational practices with fellow coders (Brennan & Resnick, 2012). Developing a *connecting* perspective involves investment and belief in the idea that coding practice benefits from access to others' coding as well as through actual interactions with fellow coders, in person and through online coding communities (Brennan & Resnick, 2012; Resnick et al., 2009). In the inclusive classroom, this might manifest as moves made toward encouraging independent collaboration, review, and sharing of projects with other coders. Brennan and Resnick (2012) describe the value of *connecting* as an opportunity to create with others, and in doing so, increase the coder's ability to create more together than they might have alone.

Structural affordances of block-based programming in scratch

We couple our discussion of best practices for teaching computational concepts, practices, and perspectives with an overview of how Brennan and Resnick (2012) framework structurally affords differentiation of instruction for diverse learners through its use of Scratch: a media rich block-based programming environment designed for novice programmers (Resnick et al., 2009). Implementing block-based programs in conjunction with sound instructional practice is considered an effective strategy for developing students' CT skills (Grover et al., 2015). Our case study teachers used Scratch to teach a CT curriculum (see Table 4), provide a rich context for actively cultivating students' CT and facilitate opportunities for conversation about CT.

Scratch's structural affordances also provided context for examining inclusive instructional moves through its low floor and high ceiling (Resnick et al., 2009). Low floor allows for tasks that learners of diverse ability levels can access while high ceiling provides extended learning opportunities (Resnick & Silverman, 2005; Wolz et al., 2009). Additionally, Scratch's block-based programming provides wide walls and scaffolding for incrementally structuring learning while presenting advanced computational constructs for proficient coders (Resnick et al., 2009). Finally, Scratch's wide walls provided multiple pathways coders of diverse ability levels could take while learning (Resnick et al., 2009). As a result, students in our case study were able to practice assembling programs in Scratch without getting bogged down in the syntactical requirements of traditional programming languages (Weintrop & Wilensky, 2018).

Finally, it is important to note that while Scratch as an inclusive platform has many affordances related to cognitive accessibility as described above, it is still inaccessible to students with visual impairments. The visual nature of programming environments, including Scratch, can make it difficult for visually impaired students to engage preferred interaction modalities (e.g., voice input, see Branham & Roy, 2019). Thus, our study of inclusive practices using Scratch is limited to students that do not present with visual

impairments. Future integration of accessible interfacing for programming environments such as Scratch could include extended audio, voice activation, and "speech to code" commands – features essential for users with visual impairments and broadly popular among users with and without other forms of disability (Branham & Roy, 2019; Pradhan et al., 2018).

Methodology

This case study is situated within a broader collaborative network of university and K-12 researchers and practitioners seeking to promote CT for diverse students in Grades 3–5 using a CS curriculum integrating Scratch (see Table 4). This federally funded network functioned as a research–practice partnership, guided by the principles of Design-Based Implementation Research (DBIR), between a public research university and a school district in Southern California. The network was tasked with designing CS education interventions to implement, study, and refine – the goal being to cultivate student academic outcomes and interest in CS through programs that emphasize integration of CT and language and literacy development.

Using a comparative case study design integrating multiple sources of evidence to investigate contextual conditions (Yin, 1994), we examine the diverse ways two fourth-grade teachers, Esther and Carla, and their students learn to code with Scratch at one of our partner schools within the network. As a goal of our partnership was to develop materials that could feasibly be used by all elementary teachers in the district with a reasonable amount of professional development, the initial piloting was done with eight veteran teachers who had extensive background and experience teaching CS to diverse elementary students.

We specifically focus on instruction in Esther and Carla's inclusive general education and GATE classrooms because of their large number of learners with exceptionalities. Esther and Carla teach at a public elementary school primarily serving students of Latino descent (96%), a significant number of which are second language learners (63%), eligible for free or reduced-price lunch (91%), and students with disabilities (13.6%). Both Esther and Carla were successful early adopters of Scratch within the network and agreed to our observation of classroom sessions throughout the school year.

Esther teaches an inclusive general education classroom comprised of students with and without disability in which 20% of the students were identified as having mild/moderate levels of disability; including two with a primary disability designation of autism and one identified as twice exceptional with giftedness and attention deficit. A shared classroom aide was observed offering push-in support to individual students with and without disability during individual and small group activities in approximately half the observations conducted in Esther's classroom. Carla teaches an inclusive Gifted and Talented Education (GATE) class in which 65% of students have been formally identified as gifted. Students are referred to Carla's class by way of teacher recommendation, writing proficiency, grades, or achievement test scores. Both Carla and Esther are GATE certified and therefore have received training on meeting the needs of students with disabilities and students in Gifted and Talented education programs.

Our study cases were developed at the classroom level, with one case each for Esther and Carla, respectively. Development of the study cases consisted of attention to, and

subsequent analysis of, captured teacher and student moves within classroom instruction using both inductive and deductive analytic approaches (Hatch, 2002). Our primary sources of data included transcriptions of 17 audio-recorded CS classroom lessons, fieldnotes taken during classroom observations, and reflective conversations with the teachers and students after each lesson. Reflective conversations with teachers involved discussing their perceptions of what went well and what could be improved in their lesson planning. Reflective conversations with students involved discussing their programming processes and how they felt about CS learning. We paid particular attention to instances in which reflective discussions supported or contradicted observations. Our analysis of the data was informed by the authors' combined 15 years of classroom teaching experience with exceptional and diverse learners.

Development and analysis of our case studies were guided by the following questions:

RQ1: *What strategies do teachers use to engage learners with exceptionalities in CT?*

RQ2: *How do teachers' instructional moves align with CT practices and perspectives, as exemplified in* Brennan and Resnick (2012) *framework?*

To answer our first research question, a constant comparative method of analysis (Miles et al., 2013) was used to examine the strategies teachers used as they engaged students with the CT curriculum. First and second cycles of coding using an inductive, grounded theory approach (Glaser & Strauss, 1967) were used to identify four thematic categories: *strategies for learning, strategies for teaching, strategies for making connections*, and *strategies for managing behavior* (see Table 1). *Strategies for learning* refer to the moves students make to learn the content presented by the teacher and/or other students. *Strategies for teaching* are the moves a teacher undertakes to teach class content. *Strategies for making connections* include meta-cognitive moves students and teachers engage in together to generate meaning and understand complex concepts. *Strategies for managing behavior* refer to the actions a teacher might take, or ask students to do, to redirect or guide student behavior.

The first cycle of coding was used to identify all resulting codes that could pertain to our first research question. The second cycle of coding was then used to refine, consolidate, and thematically subsume these codes under the *Strategies* categories identified in the first cycle of coding (Saldaña, 2016). These resulting categories and codes were used to refine the coding framework and develop a codebook for analyzing the remainder of the classroom lesson transcriptions. Data matrices (Miles et al., 2013) were then used to cluster patterns (Saldaña, 2016) and visually represent the most frequently occurring *Strategies for Teaching, Learning*, and *Making Connections* in Esther and Carla's inclusive general education and GATE classrooms (see Findings). These matrices provided a foundation for our analysis for Esther and Carla's instructional strategies in the teaching of CT to their students (see Findings).

To uncover how teachers' instructional moves aligned with CT practices and perspectives, in response to our second research question, we identified the most salient teacher strategies resulting from research question one and used content analysis (Hsieh & Shannon, 2005) to map how these strategies aligned with Brennan and Resnick (2012) CT framework. Directed content analysis is a sweeping deductive analytic strategy that

Table 1. Coding framework excerpt.

Categories	Strategies for Learning	Strategies for Teaching	Strategies for Making Connections	Strategies for Managing Behavior
Sample Codes and Definitions	Recalling Information *Retrieving previous information shared with class.* Using Background Knowledge *Applying previously learned content to current learning.*	Modelling Task *Demonstration to develop understanding.* Scaffolding for Understanding *Breaking down lesson activity or objective into smaller easier-to-follow steps.*	Discussing Computational Concepts *Discussing attributes of functions, commands, and coding activities needed to operate computer program.* Making Interdisciplinary Connections *Using one subject or concept to teach another (e.g., using mathematical equation building to describe coding).*	Checking-in *Attention to how students handle situations (coping), use words (speaking), manage body (body aware), actions (behavior), (time).* Redirecting Attention *Strategies used to guide students towards desired behavior (e.g., shifting attention to alternative activity).*

allows for expeditious coding of broader segments of data, in this case, classroom practice (Hsieh & Shannon, 2005).

Using a directed approach to content analysis allowed us to use the domains of CT as a framework for identifying inclusive instructional moves used to afford students access to the CT curriculum in Esther and Carla's general education and GATE classrooms, respectively (see Findings). To do this, we reviewed all classroom lesson transcripts, conversations, fieldnotes, and identified all teaching strategies that appeared to exemplify or describe one of the principal CT domains. We then subsumed these teaching strategies across the domains of Brennan and Resnick (2012) framework. Our content analysis resulted in the following thematic alignments between instructional moves and computational domains: *scaffolding* and *"Big Ideas" discussions* in Esther's classroom, and *peer feedback* and *online community participation* in Carla's classroom – to develop computational practices and perspectives (see Findings).

To ensure trustworthiness, we used multiple strategies to minimize researcher bias and address reliability and validity concerns as they relate to our collection and analysis of data. First, we used the constant comparative method of analysis (see above) to discuss the results of first and second cycle codings with our research team to mitigate researcher bias and reliability concerns related to the development, revision, and application of a coding scheme to data (Miles et al., 2013; Sandelowski, 1993). We addressed truthfulness and validity of findings using respondent validation, in which we asked our case study participants, Esther and Carla, to review our findings and comment on whether identified themes accurately reflected their experiences in the classroom (Long & Johnson, 2000). Finally, we used multiple data sources and analytic methods (see above) to triangulate and approximate more comprehensive findings (Kuper et al., 2008).

Findings

We now present two cases – one an inclusive general education classroom comprised of students with and without disabilities, the other an inclusive GATE classroom comprised of students with and without giftedness – to answer research questions one and two, respectively, by 1) illustrating the diverse approaches teachers used to teach CT to two different groups of learners with exceptionalities and 2) exemplifying the application of a CT framework (Brennan & Resnick, 2012) to teachers' instructional moves. Findings for research question one consist of a discussion of salient strategies (frequency count ≥ 10) used within Esther and Carla's classrooms, coupled with frequency counts (in parentheses) using a table format (see Tables 2 and 3 below). Findings for research question two consist of analyzing Esther and Carla's most frequently used instructional moves in relation to CT practices and perspectives.

RQ1: What strategies do teachers use to engage learners in CT?

Strategies used in Esther's inclusive general education classroom

Esther's most frequently used strategy for teaching was scaffolding for understanding (see Table 2). Scaffolding for understanding, which refers to the different ways Esther broke down a lesson or activity into smaller, easier-to-follow steps, was critical to creating access to the Scratch curriculum for her learners with exceptionalities, several of whom

Table 2. Frequently used strategies in esther's inclusive general education classroom.

Strategies for Learning	Strategies for Teaching	Strategies for Making Connections	Strategies for Managing Behavior
Choral response (29) Paying attention (19) Offering suggestions (11) Problem solving (11)	Scaffolding for understanding – verbally (43), physically (12), visually (23), acting out (3), graphically (2) Prompting response (58) Asking questions (54) Providing instructions (33) Giving commands (33) Defining key concepts (29) Displaying info – visually (16), verbally (2), graphically (2) Calling for volunteers (17)	Discussing "big idea" (40) Using think-alouds (38) Discussing computational concepts – algorithms (7), circuits (7), coding (5), sequences (4), drivers (5), programming (3) Making inquiries (24) Activating knowledge (20) Comprehension check (11) Providing rationale (10)	Checking-in – body-aware (15), time (13), behavior (14), coping (5), speaking (4) Request to – self-regulate (3), engage (14), be silent (7), summarize desired behavior (4) Reinforcement – praise (9), consequence (8), warning (4) Redirecting attention (20)

required information to be presented in an incremental and recursive manner. As such, a defining character of Esther's teaching was an increased intensity, frequency, and duration with which she scaffolded instruction. Whereas a teacher might typically scaffold the more difficult aspects of a lesson as the need arises, Esther's scaffolding was premeditated and extended through all components of her lesson implementation. In Esther's class, scaffolding supported students' engagement with CT concepts, taking advantage of strengths in their systematic approaches to coding and persistence in developing CT. Esther scaffolded the lesson content using primarily verbal, physical, and visual prompting (see Table 2). Verbal prompts included use of vocal tone, volume, and expression to accentuate/punctuate key points in activity instructions and connected text. Physical prompts included acting out computational concepts. Visual prompts included use of video, PowerPoint, and graphic organizers to visually represent key concepts, vocabulary, steps, and instructions.

Esther's classroom also spent a significant amount of class time using strategies for making connections to discuss computational concepts. Esther made it a point to connect discrete computational concepts to a broader "big idea" or lesson purpose; this was primarily done through Esther's use of think-alouds, which we define as the act of verbalizing our thinking as a method for monitoring comprehension (see Table 2). Discussion of computational concepts through use of think-alouds was central to Esther's instruction and focused on the class discussing the attributes of different kinds of functions/commands/coding needed to operate the Scratch programming application.

Because Esther's students required extensive amounts of scaffolding and prompting, it makes sense that the most common strategies for learning included choral response to Esther's directives and paying attention to Esther's direct instruction (see Table 2). Choral response, a verbal repetition strategy in which students respond in unison when prompted with a cue, promotes an increase in active student response through the modeling of appropriate responses and provision of "thinking pauses" – all helpful supports for students with exceptionalities (Heward et al., 1989). Choral response was an intentional strategy used by Esther to ensure student attention, engagement, and participation with the Scratch curriculum. Esther's use of choral response was most frequently observed in support of developing students' vocabulary and key concept

knowledge, and as a comprehension check of classroom activity directives. A distinguishing aspect of Esther's use of choral response, in relation to what we might observe in other general education settings, was its continual use throughout the lesson often coupled with physical prompting (see above).

Finally, Esther was observed to focus a substantial amount of time on strategies for managing behavior (see Table 2) in response to her students' needs. These supports appeared broadly useful in helping the entire class remain on task; and they were particularly crucial for Esther's students with a primary disability designation of autism and attention deficit. These students appeared to particularly benefit from behavioral supports in navigating common classroom distractors and peer interactions in support of attending to classroom activities and tasks. Esther's strategies consisted primarily of conducting check-ins to monitor her students' engagement with the Scratch curriculum. Checking-in on students occurred in various areas, including how they manage their body (body awareness), manage their time, handle a situation (coping), use their words (speaking), and manage their learning and behavior (self-regulate) (see Table 2). These strategies supported students' attending, following multi-step directions, and self-regulation – facilitating increased access and participation with the Scratch curriculum.

Strategies used in Carla's inclusive GATE classroom

Strategies for teaching used by Carla principally consisted of providing one-to-one assistance and instructions to individual students throughout each lesson unit, peer-directed computing to support learner-driven problem-solving strategies during pair and small group work, and independent exploration with the Scratch curriculum during individual work time. Uses of these teaching strategies aligned with Carla's goal of positioning herself as a facilitator of her students' independent and peer-centered learning.

Carla's strategies for making connections included requesting that her students provide rationales when proposing possible solutions together as a class, in small groups, or in pairs (see Table 3). This strategy was primarily observed during the introduction of new skills, as well as during testing and debugging, and lent itself to Carla's goal of extending her students' CT skill sets and developing her students' nimbleness with the curriculum. Requests for rationales were also used as a comprehension check by Carla to ensure that her students accurately understood key CT concepts. Use of this strategy promoted the development of an exploratory approach to instruction in which learners engaged with

Table 3. Frequently used strategies in Carla's inclusive GATE classroom.

Strategies for Learning	Strategies for Teaching	Strategies for Making Connections	Strategies for Managing Behavior
Problem solving (11) Choral response (11) Clarifying request (10)	One-to-one assistance (58) Providing instructions (40) Asking questions (19) Giving commands (14) Giving multiple options for participation (13) Highlighting key info (11)	Providing rationale (22) Comprehension check (14) Discussing "big idea" (14) Making inquiries (12) Activating knowledge (12)	Checking-in – behavior (13), time (9), body-aware (2) Request to – be silent (17), engage (7) Reinforcement – reward (8), praise (6), consequence (3) Using code word (11)

Table 4. CT curriculum sequence integrating scratch.

Unit	CS Objectives	Lesson
1: Algorithms	(1) Engage in exploratory, hands-on experience with Scratch	1: Introduction to Scratch
	(2) Use pair programming to complete collaborative tasks	2: Pair Programming
	(3) Post comments to Scratch projects	3: Peer Feedback
	(4) Create an algorithm	4: Explaining Algorithms
		5: 10 Blocks Challenge
2: Events	(1) Learn about events in unplugged lesson	6: Introducing Events, Part 1
	(2) Extend knowledge of events, algorithms, and sequences by producing creative project in Scratch	7: Introducing Events, Part 2
	(3) Critique peer work and reflect on personal progress in "About Me"	8: About Me, Planning
		9: About Me, Building
3: Electricity	(1) Gain introduction to electricity by using Makey Makey devices	11: Introducing Makey Makey
	(2) Experiment to discover conductive materials with Makey Makey	12: Makey Makey Showcase
	(3) Explain circuits using principles of conductivity	
4: Loops	(1) Learn about loops in unplugged lesson	13: Introducing Loops, Part 1
	(2) Extend knowledge of loops by producing creative project in Scratch	14: Introducing Loops, Part 2
		15: Building a Band, Planning
		16: Building a Band, Building
		17: Building a Band, Showcase
5: Storytelling	(1) Explore synchronizing interactions between sprites in Scratch	18: Characters
	(2) Create a Scratch project with multiple scenes to experiment using different backdrops	19: Conversation
	(3) Extend knowledge of events, algorithms, sequences, loops, and synchronization by producing creative project in Scratch	20: Scene
		21: Creature Construction

material without the level of incremental scaffolding that might typically be observed in other general education settings; a distinguishing aspect of Carla's requests for rationales.

Given the class's interest in independent exploration, a trait common in GATE classrooms (see Worrell et al., 2019), the most frequent strategies for learning used by Carla's students included problem-solving independently and in peer groups during student-led interactions with the Scratch interface (see Table 3). Students' use of choral response and Carla's requests for clarification were also used to facilitate problem-solving. For Carla's students, these strategies supported strengths in exercising their developing independence and exploration with computing, fostering student agency as the class aimed for a high ceiling in their use of Scratch.

Similar to Esther, Carla's primary strategies for managing student behavior focused on conducting behavior and time management check-ins to monitor her students' engagement with Scratch. Carla's behavior strategies also focused on making requests of her students to be silent and engage in independent work. Carla used classroom code words to prompt specific behavior – for example, the code word "Stoplight" – to signal a shift to a desired student behavior (see Table 3). Similarities in behavior management strategies could be partly due to Esther and Carla's shared need to maintain a structured environment for engaging with Scratch, which was a novel CT curriculum for both classrooms.

Applying a CT framework to identify inclusive instruction in diverse contexts

We will now shift from our review of salient strategies used in Esther and Carla's classrooms to an exploration of Esther and Carla's instructional moves in relation to CT practices and perspectives from a prominent CT framework (see Brennan & Resnick, 2012). The purpose is to develop a better understanding of the instructional moves teachers could use to include students with exceptionalities in a CT curriculum (see Table 4) across diverse classroom contexts, in this case, inclusive general education and GATE classrooms.

RQ2: How do teachers' instructional moves align with CT practices and perspectives, as exemplified in Brennan and Resnick (2012) framework?

Teaching CT in Esther's inclusive general education classroom

Using scaffolding to develop computational practices (Being Incremental and Iterative). In this section, we describe how Esther uses scaffolding strategies to develop students' ability to be incremental and iterative in their computational practice. Scaffolding, which we define as the breaking down of an activity or objective into smaller, easier-to-follow steps, allows Esther to simplify complex computational concepts and promote understanding. Scaffolding includes a constellation of teaching strategies, including modeling, feedback, prompting, and questioning, that incrementally support students' learning within their zone of proximal development – the space between what students can and cannot do independently (Berk & Winsler, 1995; Sanders & Welk, 2005; Vygotsky, 1934–1986). Teaching strategies that scaffold student understanding allow Esther to focus her teaching on the process of learning about computational subjects – in this case, through the practice of students' being incremental and iterative about their computational work. Being incremental and iterative in Esther's case refers to her positioning

students' computational practice as an adaptive process comprised of approaching prospective solutions in multiple small steps (Brennan & Resnick, 2012).

In Esther's class this process presents as iterative cycles of whole-class discussion focused on choral response, using verbal prompting and repetition, to build student understanding of CT concepts. As previously discussed, choral response is a learning strategy in which students respond in unison when prompted with a cue to increase modeled student engagement with curricular content (Heward et al., 1989). We define verbal prompting and repetition as the provision of verbal cues aimed at soliciting specific student responses in an iterative manner to promote comprehension (Waugh et al., 2011).

Engaging in choral response inclusively connects students of multiple ability levels with the Scratch curriculum through shared interaction modeled by Esther. In the following excerpt, Esther intentionally replaces the "what" from each sentence clause with a silent pause to get her students to state the properties of sequences as she develops her explanation of algorithms (see Unit 1: Lesson 4, Table 4). Esther uses choral response to present computational actions as a series of incremental iterative steps through repetition, call, and response. Once her students get into a pattern of call and response, Esther builds upon their understanding by shifting the structure of the last sentence to include an additional computational concept. Here Esther starts with "what?" – prompting students to articulate the noun "code" to complete the clause:

Esther: *You have to be precise. They have to be in . . . [pause]*

Students: *Order!*

Esther: *They have to be? . . . [pause]*

Students: *In order!*

Esther: *And they have to be . . . [pause]*

Students: *Precise*

Teacher: *What has to be? . . . [pause]*

Students: *The code!*

Use of choral response and repetition as inclusive strategies for scaffolding learning allows Esther's students to safely test their understanding of computational concepts, prior to creating their own algorithms, in shared interactions that are digestible and incremental. Esther's approach creates access to the Scratch curriculum through sequential introduction of computational concepts and encourages students to take an active role in cultivating and constructing their learning experiences in the classroom.

Esther's heavy use of scaffolding is essential to accommodating the diverse ability levels present in her class and ensuring that students are afforded multiple iterative opportunities to access the Scratch programming environment. Esther often uses physical (e.g., acting or gesticulating), verbal (e.g., prompting, questioning, and elaborating), and visual (e.g., use of images and text) scaffolds to model and demonstrate how to solve computational problems. Esther's use of scaffolds promotes her students' CT and creates incremental movement toward a stronger understanding of complex computational concepts. In the following excerpt, Esther uses physical scaffolds, acting like a robot

moving through a series of steps to denote sequence, to augment students' learning and guide them through an exploratory hands-on experience with Scratch (see Unit 1: Objective 1, Table 4). Acting out, using an animated voice, and reciting steps are all ways Esther helps her students access the curriculum:

Esther: *Look at what I put in my picture. Can you figure out what this is?*

Student: (Guesses) *Toothpaste!*

Esther: *Okay, I'm going to read it to you because it's hard to see.*
[Esther recites steps using animated voice and body language resembling a robot]

Esther: *Step 1: Fetch toothbrush*
[students laugh]

Esther: *Step 2: Fetch toothpaste. Step 3: Put toothbrush in mouth*
[continued laughter from students]

Esther's use of scaffolding requires her students to engage physically and verbally with computational concepts, as in explaining algorithms above. In another conversation, Esther pointed out to us that because she has students with disabilities receiving special education services, it was very important that she includes sensory movement in her teaching. Esther's commentary aligns with the incorporation of CT-specific instructional supports to promote students' understanding, production, and retention of content (Snodgrass et al., 2016.)

Using "Big Ideas" discussions to develop computational perspectives (Questioning)

We now turn to discussing how Esther uses whole-class guided discussion to develop and extend her students' computational perspectives. Computational perspectives reflect the evolving understanding students hold of themselves and others in relation to the technological world around them (Brennan & Resnick, 2012). In Esther's class, these perspectives are evident in the shifts in student understanding that take place as Esther engages in whole-class discussion of computational concepts and practices.

Esther uses a questioning perspective to tap into her students' pre-existing knowledge base. Questioning, as conceptualized by Brennan and Resnick (2012), involves using strategies to help students "feel empowered to ask questions about and with technology" to expand their sense-making abilities (p. 11). In Esther's inclusive classroom, questioning cultivates students' ability to make interdisciplinary connections between pre-existing funds of knowledge and novel lesson content. We define interdisciplinary connections as those that combine or involve multiple disciplines or fields of study (Gentzler, 2003). Funds of knowledge presume student competence, place value on life experiences as sources of knowledge, and seek to capitalize on the knowledge that students already possess (Gonzalez et al., 2005).

Questioning is realized in Esther's whole-class discussions through her use of interrogation to connect students' funds of knowledge to computational concepts and practices. This interrogation results in the development of new perspectives in how students see everyday objects and events, and whose interdisciplinary nature lends itself to the formation of what Brennan and Resnick (2012) call a "programmer's mind" where students can apply CT skills toward an improved understanding of everyday computational processes.

Esther interrogates through whole-group discussion what we call "Big Ideas." Discussion of "Big Ideas" is co-constructed by Esther and her students as a strategy for making interdisciplinary connections between funds of knowledge and computational concepts and practices to engage with the Scratch programming environment and highlight key topics in the curriculum. The following excerpt demonstrates the interrogation tactics Esther uses to develop a questioning perspective and help her students identify the "Big Idea" of using algorithms to solve computational problems:

Esther: *We need a big idea. Why are we doing this lesson? What is this about? Oh, is it about hockey?*

Students: *It's about a big idea!*

Esther: *What big idea? So we have a big idea, so our big idea. What is the key idea, theory, or principle?*

[Esther shows laminated diagram]

Esther: *Oh, what is it about . . . geometry words?*

Students: *No*

Esther: *Is the whole point of this lesson you learned in geometry? No. Is the whole point to learn your colors?*

Students: *No!*

Esther: *What is the big idea? What are we learning about Jamie?*

Jamie: *(How) to tell a computer what's wrong*

In subsequent discussion, it becomes clear that Esther is not trying to get her students to define an algorithm, but rather, tap into their pre-existing funds of knowledge about mathematics and language to make an interdisciplinary connection between mathematics, language, and CT as symbolic systems that students can use to identify and understand key elements to building an algorithm:

Esther: *An algorithm is a list of steps you can follow to complete a task. In math, it's the set of steps that you calculate, right? Where our algorithm is like 20 minus parentheses, four times three. Remember those kinds of things? [students interject: Oh yeah!] And remember that the algorithms are like a description, like a language, right? Those numbers and symbols are telling a story, right? Instead of using words, we use those numbers to explain and describe something.*

This more nuanced intention is realized in Esther's use of interrogative questioning during whole-class discussion. As such, Esther's interrogation reveals her instructional priority in cultivating her students' ability to make interdisciplinary connections through "Big Ideas" discussions to develop questioning computational perspectives.

Teaching CT in Carla's inclusive GATE classroom using peer feedback to develop computational practices (testing and debugging)

Shifting focus to Carla's inclusive GATE classroom, we examine how Carla teaches her students to use peer feedback to engage in the computational practice of testing and

debugging their Scratch projects. Testing is the process of finding errors in a program, and debugging is the process of fixing errors found during testing (Brennan & Resnick, 2012; Fitzgerald et al., 2008; Pea et al., 1987). Testing and debugging practices include an ability to identify problems, experiment with, and discuss both effective and problematic program scripts with others (Brennan & Resnick, 2012; Fitzgerald et al., 2008; Pea et al., 1987). Testing and debugging as computational practices are partly developed through insight designers gain from peer feedback, as well as examining the ways other designers address problems in their own programs (Brennan & Resnick, 2012).

Carla uses peer feedback as a vehicle for cultivating her students' testing and debugging skills so that they can more independently identify and manage problems in their programming. Carla's approach helps develop and refine students' understandings of how to identify and improve project strengths and weaknesses – the goal being to cultivate engagement and ability to learn from peers' work with Scratch. In the following excerpt, Carla cultivates her students' learning from peer work by calling attention to the use of linguistic sentence frames for reviewing peers' Scratch projects (see Unit 1: Lesson 3, Table 4), providing quality feedback, and elaborating on ideas:

> Carla: *They [linguistic sentence frames] work so well because they're specific. I value hearing something about what I've done. [Especially] if it's specific feedback rather than, "Oh, nice job." It's not that "nice job" isn't friendly. That's friendly. Right? But it doesn't tell me what about it was a nice job. If you're more specific, then I'm like, "Oh, okay, so I did that . . . here's an area I can improve on."*

In the example above, Carla makes visible a central purpose of peer feedback: namely, that specific feedback makes visible areas for improvement. Carla's intention is for her students to use the understanding garnered from peer feedback to inform the testing and debugging practices they employ to improve their projects.

In additional to facilitating an understanding of *how* peer feedback can be used to cultivate testing and debugging practices, Carla also engages her students in discussion of *why* they should engage in peer feedback:

Carla: *How many of you learned something from looking at someone's project?*
[multiple students raise hands]

Carla: *Very nice! Did it give you an idea for something you might want to try?*

Students: *Yes.*

Carla: *That's good. So that's the whole point of this [peer feedback]. That's one of the big components of this. You can learn from one another.*

We see from Carla's language that she is referring to a central value of peer feedback in developing students' computational practice: learning from another's work gives designers ideas "for something you might want to try" – in other words, testing, and ultimately debugging (Unit 2: Objective 3, see Table 4). Carla's discussion of peer feedback in this manner has been shown to improve student engagement and learning, critical components in the cultivation of students' ability to learn from peer work and increase engagement with Scratch (Mitchell et al., 2017).

Facilitating participation in Scratch community to develop computational perspectives (Connecting)

We will now explore how Carla cultivates her students' use of computation as a medium for connecting to the wider world. Using computation to connect with others is essential to the computational perspective that learning and creating are social practices enriched by engagement with others (Brennan & Resnick, 2012). Connecting as critical to the development of computational perspectives centers on the premise that designers are able to accomplish more in community than they could have on their own (Brennan & Resnick, 2012). Carla cultivates this perspective through promotion of students' independent at-home participation in Scratch's online community.

Carla encourages at-home participation in Scratch's online community to examine the projects of other designers outside of class – providing an inclusive means by which students with programming abilities can improve their practice beyond the confines of the classroom. In the following excerpt, Carla exemplifies how she encourages her students to practice Scratch at home while teaching students how to use Events – Scratch blocks that cause strings of code to run – to create "About Me" interactive collages (Unit 2: Lesson 9, see Table 4):

Carla: *You know you can do this at home too, right?*

Students: *Yes*

Carla: *You can get into our Scratch studio whenever you want. I want you to start getting on. If you want something to do, and you're at home and you want to get onto Scratch, look at other people's projects, not just from our studio, but you can be looking at other projects, too. And you can look inside and learn things from that.*

Carla views her lessons as time for interactive and collaborative student exploration of the Scratch programming environment. In encouraging students to practice using Scratch at home, Carla discusses the purpose of engaging with the programming environment as follows:

> Carla: *This is about you exploring and figuring things out. I know I would have to explore and figure it out. I'm not that great at this yet. Honestly, like I don't know what all the blocks do yet.*

Not only does Carla disrupt traditional views of teacher roles, she characterizes herself as a learner alongside her students, frequently making her insights as a learner visible to the class. Carla's think-alouds place value in examining and learning from the work of fellow designers for the purpose of "figuring things out." This characteristic of her approach to interacting with students, and Carla's reflection of *how* her own learning is enriched by engaging with the online Scratch community, models the cultivation of computational perspectives that support the use of computation as a medium for connecting with the greater Scratch community.

Discussion

Support for inclusive CT instruction in relation to CSforALL policy mandates

Insights from our findings on inclusive instructional strategies and practices can be used to support and inform the development of CT instruction for learners with exceptionalities. This application is in contrast to CT literature focused on *defining* CT concepts (Grover & Pea, 2013). Seminal CT frameworks take this discussion a step further to discuss *how* CT concepts might emerge as computational practice, and subsequently how this practice informs the development of computational perspectives (see Brennan & Resnick, 2012). However, what has been lacking is an explicit illustration of how specific CT frameworks (e.g., Brennan & Resnick, 2012) can be used to identify and analyze specific instructional strategies and moves that promote the development of CT – particularly, and most importantly, within inclusive classroom contexts catering to students with diverse abilities and needs (see Hansen et al., 2016; Israel et al., 2017; Santo et al., 2019).

To date, much of the discussion in the CSforALL Initiative seeks to broaden participation for students from diverse backgrounds by focusing on the importance of access to technology and curriculum – and less so on the specific instructional strategies needed to make CS instruction inclusive and effective. What has been partly lacking is an explicit demonstration, and explanation, of how CT frameworks can be used to identify and explore instructional strategies that increase inclusion and access for students with diverse ability levels and needs. In our comparative case study of Esther and Carla's inclusive general education and GATE classrooms, we had a unique opportunity to address this pressing question of how best to cultivate accessibility of CT curricula for a diverse array of learners through an examination of inclusive instructional practices.

Our work finds that a more scaffolded approach is needed to include and address the instructional needs of diverse learners. Scaffolding, coupled with more explicit-guided instruction and opportunities for independent exploration for students who are ready, is especially important when teaching classrooms composed of students with a wide array of ability levels and needs. As such, it behooves the CS education community to engage with inclusive instructional strategies that scaffold instruction and afford access and modified levels of challenge to a diverse range of abilities. These goals will become increasingly salient as we continue to move toward integrating CS education into K-12 instruction for students with differing abilities and needs situated in diversely resourced contexts.

Moreover, our findings indicate that while each teacher used different strategies to teach CT, they were both effective in getting their students to think computationally. Explicitly scaffolded teacher-led instruction and classroom discussion of "big ideas" provided Esther's students with needed structure for understanding the complex concepts inherent to computing; these instructional strategies and moves promote *being incremental and iterative*, as well as *questioning*, in students' practices and perspectives (Brennan & Resnick, 2012; Israel, Pearson et al., 2015). Conversely, self-initiated activities incorporating peer feedback facilitated independent learning, engagement, and problem-solving practices for Carla's students; these instructional strategies and moves supported cultivation of *testing and debugging* practices and development of *connecting* perspectives (Brennan & Resnick, 2012; Israel, Wherfel et al., 2015; Kafai & Burke, 2014).

The need to better understand the specific instructional strategies and moves that promote the development of CT in inclusive settings catering to students with diverse abilities and needs is crucial as mandates are developed to push the integration and accessibility of CT curricula for ALL learners (Smith, 2016). Based on prior research (e.g., Universal Design for Learning), it is expected that these kinds of inclusive instructional practices benefit additional groups of learners, including students with limited literacy, those from under-resourced school communities, and designated English learners (Barron & Darling-Hammond, 2008; Hitchcock & Stahl, 2003). These findings ultimately contribute to the CSforALL initiative (Smith, 2016) to develop the computational literacies of students from diverse backgrounds and broaden participation in CS education. Using CT frameworks to examine instructional strategies and practices affords an avenue for practitioners and researchers to identify best practices in support of initiatives that aim to increase and diversify the number of talented students from under-represented backgrounds entering fields increasingly requiring computational literacy.

Conclusion

A growing body of research on CT demonstrates benefit for students; however, little research focuses on the specific instructional moves teachers make to tailor CS curricula to the needs of students with exceptionalities' emergent CT skills (Goode & Margolis, 2011; Kelleher & Pausch, 2007; Ladner & Israel, 2016). This study contributes to the developing literature on methods for broadening participation in CS education by using a leading CT framework (Brennan & Resnick, 2012) to explore how teachers' inclusive instructional moves were leveraged to help students with exceptionalities succeed in developing CT skills, practices, and perspectives.

Our comparative case study of Esther and Carla's inclusive general education and GATE classrooms afforded a detailed view into the diverse strategies these teachers used to provide an entry point to CT for students with varying ability levels and needs. Furthermore, applying a CT framework to our examination of CS instruction in these two distinct learning environments uncovered the strategies each teacher used – scaffolding and whole-class discussion of "big ideas" for Esther, and independent peer feedback and problem-solving for Carla – to develop specific CT learning practices and perspectives (see Brennan & Resnick, 2012). This study highlights how CT frameworks can be leveraged to identify inclusive instructional strategies and practices to increase access to CS curricula for students with exceptionalities.

Acknowledgments

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

Disclosure statement

No potential conflict of interest was reported by the author(s).

Funding

This work was supported by the National Science Foundation under Grant number 1738825. The conduct of this research has been approved by the University of California, Irvine Institutional Review Board HS#2017-3675.

Notes on contributor

Yenda Prado is a PhD in Education student at the University of California, Irvine. Her research focuses on inclusive best practices in education at the intersection of language, literacy, and technology. She studies how digital technologies are used to include and engage exceptional learners, with and without disability, as readers and writers in diverse instructional settings. She has 10 years' experience as a literacy specialist and teaching English as a Second Language.

Sharin Jacob is a PhD in Education student at the University of California, Irvine. Her research interests bring together theory from the learning sciences, computer science education, and applied linguistics to examine factors that help multilingual students succeed in mastering computational thinking. She has five years' experience teaching English as a Second Language, where she taught all levels of proficiency.

Mark Warschauer is a Professor of Education at the University of California, Irvine where he directs the Digital Learning Lab. His research focuses on the uses of digital media to promote language and literacy development and academic achievement among culturally and linguistically diverse learners. He is Principal Investigator of a National Science Foundation-funded project that is developing computational thinking curriculum for diverse learners.

ORCID

Yenda Prado http://orcid.org/0000-0003-0108-5838
Sharin Jacob http://orcid.org/0000-0003-4073-4271
Mark Warschauer http://orcid.org/0000-0002-6817-4416

References

Algozzine, R., Ysseldyke, J.E. (2006). *The fundamentals of special education:A practical guide for every teacher*. Sage Publications.

Armstrong, D., Armstrong, A. C., & Spandagou, I. (2011). Inclusion: By choice or by chance? *International Journal of Inclusive Education*, *15*(1), 29–39. https://doi.org/10.1080/13603116.2010.496192

Barron, B., & Darling-Hammond, L. (2008). *Teaching for meaningful learning: A review of research on inquiry-based and cooperative learning*. George Lucas Educational Foundation.

Basham, J. D., Israel, M., Graden, J., Poth, R., & Winston, M. (2010). A comprehensive approach to RTI: Embedding universal design for learning and technology. *Learning Disability Quarterly*, *33*(4), 243–255. https://doi.org/10.1177/073194871003300403

Basu, S., Biswas, G., & Kinnebrew, J. S. (2017). Learner modelling for adaptive scaffolding in a computational thinking-based science learning environment. *User Modeling and User-Adapted Interaction*, *27*(1), 5–53. https://doi.org/10.1007/s11257-017-9187-0

Berk, L., & Winsler, A. (1995). *Scaffolding children's learning: Vygotsky and early childhood learning*. National Association for Education of Young Children Research into Practice Series.

Branham, S. M., & Roy, A. R. M. (2019, October). Reading between the guidelines: How commercial voice assistant guidelines hinder accessibility for blind users. *In Proceedings of the 21st International ACM SIGACCESS Conference on Computers and Accessibility* (pp.446–458). Pittsburgh, Pennsylvania, USA.

Brennan, K., & Resnick, M. (2012). Using artifact-based interviews to study the development of computational thinking in interactive media design. *Paper presented at annual American Educational Research Association meeting*.

Causton-Theoharis, J., & Theoharis, G. (2008). Creating inclusive schools for all students. *School Administrator*, *65*(8), 24–25. https://www.aasa.org/SchoolAdministratorArticle.aspx?id=4936

Darling-Hammond, L. (2012). *Powerful teacher education: Lessons from exemplary programs*. John Wiley & Sons.

Dudley-Marling, C., & Burns, M. B. (2014). Two perspectives on inclusion in the United States. *Global Education Review*, *1*(1), 14–31. https://files.eric.ed.gov/fulltext/EJ1055208.pdf

Edyburn, D. L. (2013). Critical issues in advancing the special education technology evidence base. *Exceptional Children*, *80*(l), 7–24. https://doi.org/10.1177/001440291308000107

Fitzgerald, S., Lewandowski, G., McCauley, R., Murphy, L., Simon, B., Thomas, L., & Zander, C. (2008). Debugging: Finding, fixing and flailing, a multi-institutional study of novice debuggers. *Computer Science Education*, *18*(2), 93–116. https://doi.org/10.1080/08993400802114508

Gentzler, E. (2003). Interdisciplinary perspectives. *Perspectives: Studies in Translatology*, *11*(1), 11–24. https://doi.org/10.1080/0907676X.2003.9961458

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

Gonzalez, N., Moll, L. C., & Amanti, C. (2005). *Funds of knowledge: Theorizing practices in households, communities, and classrooms*. Lawrence Erlbaum Associates Inc.

Goode, J., Flapan, J., & Margolis, J. (2018). Computer Science for All: A school reform framework for broadening participation in computing. In W. G. Tierney, Z. B. Corwin, & A. Ochsner (Eds.), *Diversifying digital learning: Online literacy and educational opportunity*(pp. 45-65). Johns Hopkins University Press.

Goode, J., & Margolis, J. (2011). Exploring computer science: A case study of school reform. *ACM Transactions on Computing Education (TOCE)*, *11*(2), 12. https://doi.org/10.1145/1993069.1993076

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

Grover, S., & Pea, S. (2018). Computational thinking: A competency whose time has come. In SentanceS., BarendsenE., & ShulteC. (Eds.) *Computer science education: Perspective on teaching and learning in school* (pp. 19–38). Bloomsbury Publishing.

Hall, K., Collins, J., Benjamin, S., Nind, M., & Sheehy, K. (2004). Saturated models of pupildom: Assessment and inclusion/exclusion. *British Educational Research Journal*, *30*(6), 801–817. https://doi.org/10.1080/0141192042000279512

Hansen, A. K., Hansen, E. R., Dwyer, H. A., Harlow, D. B., & Franklin, D. (2016, February). Differentiating for diversity: Using universal design for learning in elementary computer science education. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education* (pp. 376–381). Memphis, Tennessee, USA.

Hatch, J. A. (2002). *Doing qualitative research in education settings*. State University of New York Press.

Heward, W. L., Courson, F. H., & Nayaran, J. S. (1989). Using choral responding to increase active student response. *Teaching Exceptional Children*, *21*(3), 72–75. https://doi.org/10.1177/004005998902100321

Hitchcock, C., & Stahl, S. (2003). Assistive technology, universal design, universal design for learning: Improved learning opportunities. *Journal of Special Education Technology*, *18*(4), 45–52. https://doi.org/10.1177/016264340301800404

Hsieh, H. F., & Shannon, S. E. (2005). Three approaches to qualitative content analysis. *Qualitative Health Research*, *15*(9), 1277–1288. https://doi.org/10.1177/1049732305276687

Israel, M., Jeong, G., Ray, M., Lash, T. (2017). Teaching elementary computer science through Universal Design for Learning. *Proceedings of the 51st ACM Technical Symposium on Computer Science Education* (pp. 1220–1226). Portland, Oregon, USA. https://doi.org/10.1145/3328778.3366823

Israel, M., Pearson, J., Tapia, T., Wherfel, Q., & Reese, G. (2015). Supporting all learners in school-wide computational thinking: A cross case analysis. *Computers & Education*, *82*, 263–279. https://doi.org/10.1016/j.compedu.2014.11.022

Israel, M., & Ribuffo, C., & Smith, S. (2014). *Universal design for learning: Recommendations for teacher preparation and professional development* (Document No. IC-7). Retrieved from University of Florida, Collaboration for Effective Educator, Development, Accountability, and Reform Center website:http://ceedar.education.ufl.edu/tools/innovation-configurations/

Israel, M., Wherfel, Q. M., Pearson, J., Shehab, S., & Tapia, T. (2015). Empowering K 12 students with disabilities to learn computational thinking and computer programming. *Teaching Exceptional Children*, *48*(1), 45–53. https://doi.org/10.1177/0040059915594790

Jacob, S. R., & Warschauer, M. (2018). Computational thinking and literacy. *Journal of Computer Science Integration*, *1*(1), 1–19. https://doi.org/10.26716/jcsi.2018.01.1.1

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

Kelleher, C., & Pausch, R. (2007). Using storytelling to motivate programming. *Communications of the ACM*, *50*(7), 58–64. https://doi.org/10.1145/1272516.1272540

Kuper, A., Lingard, L., & Levinson, W. (2008). Critically appraising qualitative research. *BMJ*, *337*(a), 687–692. https://doi.org/10.1136/bmj.a1035

Ladner, R. E., & Israel, M. (2016). For all in computer science for all. *Communications of the ACM*, *59*(9), 26–28. https://doi.org/10.1145/2971329

Long, T., & Johnson, M. (2000). Rigour, reliability and validity in qualitative research. *Clinical Effectiveness in Nursing*, *4*(1), 30–37. https://doi.org/10.1054/cein.2000.0106

Margolis, J. (2010). *Stuck in the shallow end: Education, race, and computing*. MIT Press.

Margolis, J., & Goode, J. (2016). Ten lessons for computer science for all. *ACM Inroads*, *7*(4), 52–56. https://doi.org/10.1145/2988236

Martin, A., McAlear, F., & Scott, A. (2015). *Path not found: Disparities in access to computer science courses in California high schools*. Level Playing Field Institute. https://eric.ed.gov/?id=ED561181

Miles, M. B., Huberman, A. M., & Saldaña, J. (2013). *Qualitative data analysis: A methods sourcebook*. SAGE Publications Inc.

Mishra, P., & Yadav, A. (2013). Of art and algorithms: Rethinking technology & creativity in the 21st century. *TechTrends*, *57*(3), 10-14.

Mitchell, I., Keast, S., Panizzon, D., & Mitchell, J. (2017). Using 'big ideas' to enhance teaching and student learning. *Teachers and Teaching*, *23*(5), 596–610. https://doi.org/10.1080/13540602.2016.1218328

National Center on Universal Design for Learning. (2009). *What is Universal Design for Learning*. Retrieved from http://www.udlcenter.org

Obiakor, F. E., Harris, M., Mutua, K., Rotatori, A., & Algozzine, B. (2012). Making inclusion work in general education classrooms. *Education and Treatment of Children*, *35*(3), 477–490. https://doi.org/10.1353/etc.2012.0020

Pea, R. D., Soloway, E., & Spohrer, J. C. (1987). The buggy path to the development of programming expertise. *Focus on Learning Problems in Mathematics*, *9*(1), 5–30. □hal-00190540□

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

Pradhan, A., Mehta, K., & Findlater, L. (2018, April). Accessibility came by accident: Use of voice-controlled intelligent personal assistants by people with disabilities. *In Proceedings of the Conference on Human Factors in Computing Systems (CHI)* (pp. 1–13). Montreal, Canada.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., … Kafai, Y. B. (2009). Scratch: Programming for all. *Communications of the ACM*, *52*(11), 60–67. https://doi.org/10.1145/1592761.1592779

Resnick, M., & Silverman, B. (2005, June). Some reflections on designing construction kits for kids. *In Proceedings of the 2005 Conference on Interaction Design and Children (ACM)* (pp. 117–122). Athens, Greece.

Rose, D. H., Meyer, A., & Hitchcock, C. (Eds). (2005). *The universally designed classroom: Accessible curriculum and digital technologies*. Harvard Education Press.

Sailor, W., & Roger, B. (2005). Rethinking inclusion: School wide applications. *Phi Delta Kappan*, *86*(7), 503–509. https://doi.org/10.1177/003172170508600707

Saldaña, J. (2016). *The coding manual for qualitative researchers* (3rd ed.). Sage Publications Inc.

Sandelowski, M. (1993). Rigor or rigor mortis: The problem of rigor in qualitative research revisited. *Advanced Nursing Science*, *16*(2), 1–8. https://doi.org/10.1097/00012272-199312000-00002

Sanders, D., & Welk, D. S. (2005). Strategies to scaffold student learning: Applying Vgotsky's zone of proximal development. *Nurse Educator*, *30*(5), 203–207. https://doi.org/10.1097/00006223-200509000-00007

Santo, R., DeLyser, L. A., Ahn, J., Pellicone, A., Aguiar, J., & Wortel-London, S. (2019, February). Equity in the who, how and what of CS education: K12 school district conceptualizations of equity in 'CSforALL' initiatives. In *2019 Research on Equity and Sustained Participation in Engineering, Computing, and Technology (RESPECT)* (pp. 1–8). Minneapolis, Minnesota, USA.

Smith, M. (2016, January). *CSforALL*. [Web log comment]. https://obamawhitehouse.archives.gov/blog/2016/01/30/computer-science-all

Snodgrass, M. R., Israel, M., & Reese, G. C. (2016). Instructional supports for students with disabilities in K-5 computing: Findings from a cross-case analysis. *Computers & Education*, *100*, 1–17. https://doi.org/10.1016/j.compedu.2016.04.011

Tissenbaum, M., Sheldon, J., & Abelson, H. (2019). From computational thinking to computational action. *Communications of the ACM*, *62*(3), 34–36. https://doi.org/10.1145/3265747

Vogel, S., Santo, R., & Ching, D. (2017, March). Visions of computer science education: Unpacking arguments for and projected impacts of CS4All initiatives. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on CS Education* (pp. 609–614). Seattle, Washington, USA.

Vygotsky, L. (1934–1986). *Thought and language*. MIT Press.

Waugh, R. E., Alberto, P. A., & Fredrick, L. D. (2011). Simultaneous prompting: An instructional strategy for skill acquisition. *Education and Training in Autism and Developmental Disabilities*, *46*(4), 528–543. https://www.jstor.org/stable/24232364

Weintrop, D., & Wilensky, U. (2018). How block-based, text-based, and hybrid block/text modalities shape novice programming practices. *International Journal of Child-Computer Interaction*, *17*, 83–92. https://doi.org/10.1016/j.ijcci.2018.04.005

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

Wolz, U., Leitner, H. H., Malan, D. J., & Maloney, J. (2009). Starting with scratch in CS 1. *In Proceedings of the 40th ACM technical symposium on CS education* (pp. 2–3). Chattanooga, Tennessee, USA.

Worrell, F. C., Subotnik, R. F., Olszewski-Kubilius, P., & Dixson, D. D. (2019). Gifted students. *Annual Review of Psychology*, *70*(1), 551–576. https://doi.org/10.1146/annurev-psych-010418-102846

Yin, R. K. (1994). *Case study research design and methods: Applied social research and methods series*. Sage Publications Inc.