The CATESOL Journal

Sharin Rawhiya Jacob
Digital Promise Global

Mark Warschauer
University of California, Irvine

Elementary Computing for All: A Computational Thinking Curriculum for Multilingual Students

Over the last decade, there has been an explosion of national interest in computer science (CS) education. In response to this, several organizations and initiatives have emerged in recent years to expand the CS pipeline. However, within these broad and laudable efforts, one important area has been largely overlooked—the instruction of CS to multilingual students, including the large and growing number of students designated as English learners in K-12 schools. These are one of the most underserved and understudied groups in CS education. In this article, we draw on existing research, as well as our own and others' theoretical and empirical work to date, to put forth both a framework and curriculum for teaching CS to multilingual students.

Keywords: computational thinking, computer science, English learner, multilingual, curriculum, elementary

Introduction

Over the last decade, there has been an explosion of national interest in computer science (CS) education. The growing importance of information technology professionals, the reliance of many sciences on computational methods, the influence of social media on various aspects of life, and international competition in artificial intelligence underscore the urgency to transform the US education system. This transformation is increasingly convincing policymakers and the general public to integrate comprehensive computer science education for all in K-12 classrooms. Despite the growing potential of CS in K-12 contexts, these technologies are not accessible to all students.

In response to the need to broaden participation in CS, several organizations and initiatives have emerged in recent years to expand the CS pipeline. An early and important concern was the lack of female participation in CS (Cheryan et al., 2009; Kelleher & Pausch, 2007), and several initiatives have been launched to address this issue (Beyer, 2014; Cheryan et al., 2009; Kelleher & Pausch, 2007; Zhou et al., 2020). More recently, projects and organizations have begun to tackle, through equity focused CS initiatives, other groups that are too frequently excluded from CS, especially students of color (Allen-Handy et al., 2020; DiSalvo et al., 2011). There have also been important initiatives to extend CS instruction to children with special needs (Israel et al., 2015; Ladner et al., 2021).

However, within these broad and laudable efforts, one important area has been largely overlooked—the instruction of CS to multilingual students, including the large and growing number of students designated as English learners in K-12 schools in the United States. The percentage of students designated as English learners grew from 8.1% (or 3.8 million students) in 2000 to 9.6% (or 4.9 million students) in 2016 and is projected to reach 25% of the student population by 2025 (McFarland et al., 2019). Though there is a dearth of data on this rapidly growing population's access to CS studies and careers, there is reason for considerable concern. For example, US public schools serve a large number of Latine students who are designated as English learners, a group distinctly underrepresented in the computer science (CS) arena (Code.org, CSTA, & ECEP Alliance, 2020; NCSES, 2021). As pointed out by Jacob et al. (2023), studies show that these Latine students who are designated as English learners encounter the most significant obstacles in both academic progress and accessing technological resources (Irwin et al., 2021). Moreover, there exists a widespread yet flawed notion that students designated as English learners must reach a certain level of English fluency before they can embark on CS education (Jacob et al., 2018; Jacob et al., 2023). This belief has led to a noticeable discrepancy in CS course offerings for students designated as English learners. Research demonstrates that schools with a larger presence of students designated as English learners (above 12%) provide significantly fewer CS courses compared to those with lower proportions of these students (Martin et al., 2015). This disparity is more pronounced in economically disadvantaged areas where most English learners reside, with schools in such neighborhoods offering limited CS courses (Code.org, CSTA, & ECEP Alliance, 2020). In addition, the representation of individuals from underrepresented groups in the CS workforce or media portrayals is minimal (Royal & Swift, 2016). This lack of representation, combined with limited access to technology at home and a dearth of mentors with computing backgrounds, adversely affects their interest in the CS field (Jacob, et al., 2022; Jacob et al., 2023). Finally, the instructional materials used in K-12 settings often fail to reflect the cultural and linguistic diversity of the students being served. This shortfall in culturally responsive teaching materials is an overlooked aspect in CS education as it relates to inclusiveness (Goode, 2008; Goode et al., 2018).

Because multilingual students are left at the margins of CS education, researchers, practitioners, and policymakers need to better understand how to provide culturally and linguistically responsive instruction to these students. In this article we examine the landscape of CS in K-12 education. We then discuss five empirically supported strategies for engaging multilingual students in Science, Technology, Engineering, and Mathematics (STEM) education as outlined in a recent report by the National Academies of Sciences, Engineering, and Mathematics (NASEM, 2018). We describe how these practices have been embedded in a yearlong CS curriculum designed to meet the needs of multilingual students, called Elementary Computing for All. We conclude with empirical findings synthesized from years of research on our Elementary Computing for All project on student and teacher outcomes from a yearlong implementation of the Elementary Computing for All curriculum. We then discuss implications for future research.

CS Education in K-12

To understand why multilingual students are marginalized in CS education, it is first necessary to clarify the nature of CS education in K-12 schools, encompassing three overlapping constructs: CS, coding, and computational thinking (CT). CS represents a discipline like science, math, engineering, and medicine and refers to the study of computers, algorithms, hardware, and software and their effects on society (CITE). Coding refers to the reading and writing of computer programming languages. Some coding languages are text-based such as Java, Python, C, and C++. At the same time, many novice learners use block-based programming languages, such as Scratch and Alice, to provide entry-level access to computational methods. Most block-based languages contain a coding console or an area where students can give programming commands. The programming commands in Scratch are coded into blocks that students snap together like Legos. These programming languages also typically contain a stage, or an area in the platform, in which students can choose from or create characters, backdrops, and sounds to develop games, stories, and animations. Block-based languages are typically taught in elementary and middle school, and students shift to text-based languages when they reach high school. However, findings from two introductory CS courses at the undergraduate level indicate that block-based languages such as Scratch are effective in transitioning students to text-based languages even at the postsecondary level (Mishra et al., 2014; Sykes, 2007).

To make visible the problem-solving strategies inherent in CS and coding for students in K-12, Jeanette Wing (2006) coined the term "computational thinking" to refer to a problem-solving approach that uses concepts essential to computing but that can be generalized to a wide array of disciplines. Although computational thinking can be learned through coding, it can also be learned through unplugged activities that teach computational thinking without the use of computers. Computational thinking involves formulating questions in a manner that can be interpreted by a computer to achieve desired results (Wing, 2006). Fundamental to computational thinking is abstraction, which involves the ability to navigate multiple layers of a problem by identifying essential components of a model and ignoring extraneous details. In addition to abstraction, computational thinking skills include but are not limited to automation, algorithmic thinking, modularization, and data analysis (International Society for Technology in Education & CS Teachers Association, 2011).

There is no consensus on the definition of computational thinking (Barr & Stephenson, 2011; Grover & Pea, 2013). Traditionally, CT has been thought to involve key computational concepts (i.e., sequence or order, conditionals or if-then statements, loops, or repeat blocks; Grover & Pea, 2013). Others argue that focusing on concepts represents a limited view of computational thinking and point to the practices and perspectives used (Brennan & Resnick, 2012). Some researchers have taken a sociocultural perspective, focusing on how CT can be used to address broader social issues such as equitable participation (Erete et al., 2017). Additionally, scholars have drawn upon Papert (1980) and diSessa (2000) to describe how CT connects to powerful ideas in the STEM disciplines (Wilensky et al., 2014). However, all the aforementioned researchers agree that CT should be taught to all students in K-12, as early as elementary school (Jacob & Warschauer, 2018).

To this end, it is widely agreed that learning computational methods provides several affordances for diverse learners. First, they are critical for advancing scientific, technical, and social innovation. CS can be used to solve broader societal problems, such as tracing the path of infectious diseases or providing warning systems for disastrous hurricanes. At the same time, students can use CS to solve problems in their own communities (Tissenbaum et al., 2019). For example, students in desert climates integrated arts and crafts with computing technologies to develop drought- and heat-resistant lunch boxes (Ball et al., 2017). The second affordance to teaching CS involves developing digital citizenship by preparing students to explore ethical considerations around computing and actively participate in digital media creation (Vogel et al., 2017). For example, students can discuss issues related to racial bias in artificial intelligence-driven facial recognition algorithms or examine issues related to copyright with the advent of generative artificial intelligence software such as ChatGPT. Third, the thought processes underlying CS are applicable to a rapidly increasing number of jobs that grant power and prestige in a knowledge economy.

Recently, CS educators have shifted the focus on learning from job preparation to the promotion of student agency (Vogel et al., 2017). Early exposure to high-quality computing education provides students with experiences that equip them with the means to decide whether to enter the computing workforce (Vogel et al., 2017). A panel of educational stakeholders has also pointed to the role of social justice in CS education (Vogel et al., 2017). When students take critical approaches to computing, they are provided opportunities to acknowledge bias in technology while providing meaningful critiques of the discipline (Vogel et al., 2017). Finally, the ability to think computationally has been positioned as a fundamental literacy that all students should acquire to fully participate in today's society (diSessa, 2000; Jacob & Warschauer, 2018; Wing, 2006). Given these constellating factors, it is critical that we engage culturally and linguistically diverse learners in CS education so that they can benefit from the multiple advantages of its worldly applications.

Engaging Multilingual Students in CS

Borrowing from research on STEM education, multilingual students' CS learning has been supported by several empirically supported practices. For example, building on multilingual students' existing resources can increase their participation in CS (Jacob et al., 2022). We draw from a recent report by the National Academies of Sciences, Engineering, and Mathematics (NASEM, 2018), which uncovers five effective practices for engaging multilingual students in STEM: (1) engage students in disciplinary practices; (2) encourage rich classroom discourse; (3) build on students' multiple meaning-making resources; (4) encourage students to use multiple registers and modalities; and (5) provide explicit focus on how language functions in the discipline. Although research on the efficacy of these practices in math and science classrooms has been well established (Jona et al., 2014; Sengupta et al., 2018; Weintrop et al., 2016), studies on how these practices fit within the field of CS are limited. Below, we provide an overview of how these practices can be applied to CS education. We borrow heavily from empirical studies on Elementary Computing for All to discuss these five practices (Jacob, Parker, Warschauer, 2022; Jacob et al., 2022).

In the CS classroom, students **engage in disciplinary practices** by testing theoretical conjectures about the empirical world. Several studies have demonstrated the affordances of engaging English learners (ELs) in inquiry-based instruction in S (Estrella et al., 2018; Janzen, 2008; Rosebery et al., 1992). Notably, scientific inquiry focuses on identifying and evaluating hypotheses to understand a set of principles governing the physical world (Giere, 1999). This objective contrasts with inquiry in CS, which focuses on constructing and testing logical processes to address an abstract computational problem (Wing, 2006). Instruction in CS then seeks to develop a set of skills, practices, and dispositions essential to understanding computational methods (Wing, 2006). Although inquiry-based learning represents a promising approach to engaging multilingual students in CS, several studies have shown that it is the highly structured and scaffolded approaches to inquiry-based instruction that improve social and academic outcomes for diverse learners (Kirschner et al., 2006; Reiser, 2004; Sweller & Cooper, 1985). Inquiry-based learning that lacks the appropriate scaffolding and teacher modeling often results in students getting stuck, which discourages them and leads to lost opportunities (Bransford et al., 2009).

The second effective practice for engaging multilingual students in STEM involves **building on students' existing resources** (NASEM, 2018). This can be achieved by leveraging students' linguistic, cultural, semiotic, and embodied resources to increase their participation in disciplinary practices (Brown & Ryoo, 2008; Vogel et al., 2020). As students negotiate meaning in CS classrooms, they have opportunities to make connections between teacher output and computational artifacts, problematize knowledge, and question misconceptions (Gibbons, 2007).

Collaborative activities that **encourage rich classroom discourse and interaction,** such as pair work, group work, and peer feedback, facilitate opportunities for growth in both language and CS learning. When students are provided multiple opportunities to collaborate in CS classrooms, they engage in active listening as well as productive language functions to construct meaning, such as describing complex problems, identifying possible solutions, testing and debugging solutions, and analyzing program efficiency (Jacob et al., 2018; Swanson et al., 2014). In particular, scaffolding collaborative discourse has been shown to engage language learners in STEM practices (Jadallah et al., 2011). During scaffolded collaborative talk, students are provided multiple opportunities to use their everyday sense-making abilities to practice the functions of social interaction, such as paraphrasing, asking for clarification, building on others' ideas, probing for meaning, and engaging in meaningful discussion (Michaels & O'Connor, 2015). Shea and Shanahan (2011) found that integrating language and disciplinary learning in a manner that facilitated peer-to-peer talk increased both English Language Arts (ELA) and math scores for multilingual students. Multilingual students are those who speak or are learning more than one language, including students who are designated as English learners. We refer to multilingual students throughout, as it represents an asset-based term for students who speak more than one language, whereas the term English Learner focuses solely on the target language.

In addition to collaboration and inquiry, hands-on activities, visualizations, and scaffolding strategies **engage students in multiple modalities** to better teach key language and concepts. This is evidenced by Ryoo and Bedell's (2019) study on the value of visualizations for helping multilingual students understand science content. Research indicates that expanding beyond linguistic modalities of speech and text to include nonlinguistic modalities such as symbols, simulations, models, and graphical notation facilitates the learning of key computational concepts (cf. Grapin et al., 2021; Lee et al., 2019). In CS classrooms, visual representations such as models and simulations are frequently used to engage students in computational methods. These representations leverage students' semiotic resources for CS learning (Grapin et al., 2021).

Finally, linguistic scaffolds **provide explicit focus on how language functions in the discipline**. Early research situated in sociology and anthropology proposed that science is learned through social practices in which language learners construct knowledge through scientific discourse (Kelly & Chen, 1999; Latour, 1987; McGinn & Roth, 1999). Although scientific processes lend themselves to language functions (i.e., analyze, classify, demonstrate, hypothesize, observe, record, strategize, summarize, etc.; Lemke, 1990; Snow et al., 1989), little work has been done to explore the functional language of CS. Jacob et al. (2018) described the first effort to develop CS language functions that provide linguistic scaffolding for the development of computational thinking. In CS, these functions include discussing the relationship between desired outcomes and coding strategies that result in those outcomes and analyzing errors in coding that prevent those outcomes from being achieved (Jacob et al., 2018). Explicit teaching of the corresponding language forms and functions reinforces students' understanding while developing their linguistic repertoires (NASEM, 2018). For example, elementary students in a program that integrated content, English Language Development (ELD) standards, and corresponding linguistic frames into inquiry-based STEM unit plans showed significant increases in ELD and ELA scores compared to students in a traditional elementary science curriculum (Zwiep & Straits, 2013).

Other researchers focus on the theory of translanguaging (García & Li, 2014) to combat the idea that coding is an exclusive discourse (Vogel et al., 2020). They argue that viewing coding in this manner has excluded marginalized groups, such as multilingual students, from participating in CS (Vogel et al., 2020). Translanguaging refers to the leveraging of students' entire linguistic repertoires for learning and represents a mechanism for including marginalized students in computing discourse (García & Li, 2014; Vogel et al., 2020). Findings from a study examining teacher use of translanguaging to engage multilingual students in CS education indicated that students leveraged their varied resources (i.e., linguistic, cultural, semiotic, and embodied) to develop computational literacies and increase their participation in CS learning (Vogel et al., 2020).

Elementary Computing for All Curriculum

Beginning with a partnership between the University of California, Irvine (UCI) and Santa Ana Unified School District (SAUSD), Elementary Computing for All is a Research Practice Partnership that has since included two large urban school districts and a dual immersion public charter school in Santa Ana. Almost 95% of students in each of these three partner school districts are Latine, and roughly 50% of elementary children are classified as ELs. Chicago Public Schools and Boston Public Schools have also adopted the curriculum. Hundreds of unique users from 30 states and 24 countries have accessed the curriculum on our project website, and teachers from numerous districts and several other countries have joined.

Research Practice Partnerships (RPPs) are collaborations between researchers and practitioners that seek to identify and solve problems of practice, or issues that participating practitioners consider to be of central importance (Coburn et al., 2013). RPPs use a bottom-up approach in which knowledge, although grounded in sound theory, is generated from the lived experiences of participating practitioners. Solving issues that are of high priority to practitioners while also engaging them in research and development increases the sustainability of implementation over traditional reform-based research interventions (Eisenhart & Towne, 2003).

During the beginning stages of the partnership, SAUSD proposed a goal of increasing both STEM and literacy learning in their district. The Elementary Computing for All curriculum was designed to address this problem of practice by designing an integrated Grade 3-5 computational thinking and literacy curriculum in concert with teachers and administrators that meet the needs of the districts' large percentages of ELs. Although the curricular intervention has undergone several iterations, it is grounded in the five practices described above as outlined in the National Academies report (NASEM, 2018). What follows is a description of how the curriculum has been adapted to incorporate these empirically supported practices (see Table 1). The descriptions of these practices below borrow heavily from existing research on the Elementary Computing for All curriculum (see Jacob, Parker, Warschauer, 2022; Jacob, Nguyen et al., 2022).

Table 1
Aligning Elementary Computing for All with the NASEM Practices

Practices	***Methods/Techniques***	***Strategies/Activities***
Engage students in disciplinary practices	● Computational thinking skills, practices, and perspectives ● CS inquiry (5 E's) ● Standards alignment (CSTA, ELD, ELA)	● Inquiry-based learning ● Use/Modify/Create ● TIPP&SEE ● Culminating unit projects ● Unplugged activities
Engage students in productive discourse and interactions with others	● Pair programming ● Small group ● Whole class	● Scratch charades ● Pair programming culminating projects ● Peer feedback opportunities ● Unplugged activities
Utilize and encourage students to use multiple registers and multiple modalities	● Gestures ● Symbols ● Pictures ● Body movement ● Visualizations ● Opportunities for reflection ● Teacher questioning techniques	● Multiple modalities (examples) ● Unplugged activities ● Simulations ● Videos ● TIPP&SEE ● Registers (examples) ● End-of-unit reflections ● Sentence frames
Build on students' existing resources	● Culturally responsive materials ● Inductive pedagogical approaches	● Responsive storybooks ● Code.org video resources ● Unplugged activities ● Memorable mentors
Provide some explicit focus on how language functions in the discipline	● Systematic functional linguistic scaffolding ● Language objectives	● Language frames ● End-of-unit reflections ● Interaction-based language activities

Elementary Computing for All **engages students in disciplinary practices** by using inquiry-based learning to provide authentic contexts for language use through student engagement in exploration, experimentation, and hands-on activities (National Research Council [NRC], 1996). The first iteration of the curriculum integrated inquiry-based learning by borrowing from the 5 E's model of inquiry (Bybee, 1997), which includes five stages of inquiry-based instruction: Engage, Explore, Explain, Elaborate, and Evaluate. Engagement involves creating student interest in a subject. Exploration refers to immersing students in hands-on activities that help them understand scientific phenomena. Explanations involve drawing on students' linguistic and conceptual resources to provide a rationale for observed phenomena. Elaboration involves the application of the learned material. Evaluation involves student reflection on their newly obtained knowledge.

Although inquiry-based instruction using the 5 E's yielded positive results for multilingual students (Jacob et al., 2020; Jacob et al., 2022), researchers and practitioners of Elementary Computing for All sought to understand the level of structure and scaffolding necessary to maximize inquiry-based learning for multilingual students in CS. To this end, Jacob, et al. (2020) sought to examine the types of inquiry-based instruction that supported multilingual students' development of computational thinking skills and CS identities. Jacob, Nguyen, et al. (2020) used a framework created by Windschitl (2003) to come up with four types of inquiry. These types ranged from more structured and scaffolded approaches to inquiry to more open and exploratory approaches to inquiry-based instruction. They and other researchers found that students who received more structured approaches to inquiry-based instruction developed more sophisticated computational artifacts and stronger CS identities (Jacob, Nguyen, et al., 2020; Prado et al., 2022). Based on these findings, UCI and the University of Chicago partnered with SAUSD, MUSD, and Chicago Public Schools to add more structure to the yearlong computational thinking curriculum.

Elementary Computing for All applied a well-studied strategy from reading research, THIEVES (Manz, 2002), to students' CS projects to develop the TIPP&SEE strategy (Salac et al., 2020). Furthermore, to provide students with exposure to key CS concepts prior to creating projects on their own, Elementary Computing for All integrated the Use→Modify→Create model (Lee et al., 2011) into curricular units. What follows is an explanation of the Use→Modify→Create model (Lee et al., 2011) and TIPP & SEE (Salac et al., 2020).

Research suggests that providing unstructured inquiry with minimal guidance to diverse learners can lead to frustration and wasted opportunities (Bransford et al., 2009). Instructional models such as Use→Modify→Create (I. Lee et al., 2011) provide the necessary structure by having students first *use* existing programs, then work together to *modify* them, and finally *create* their own. The Use→Modify→Create model has shown promising results through its integration into a wide array of CS curricula (Franklin et al., 2020). During the *use* stage, an additional layer of scaffolding based on the reading comprehension strategy THIEVES (Manz, 2002) can be added to CS activities called TIPP&SEE. TIPP&SEE is designed to leverage students' metacognitive strategies to prepare them for learning complex computational concepts. In the first stage of TIPP&SEE, students read the *Title* of the program and make a prediction about its contents, then analyze its *Instructions* to better understand the task at hand, and finally identify the *Purpose* of the program to consider the most salient features of learning. Finally, they *Play* with the program to understand its features and document their observations. Students then turn to the program's code, by first examining the Sprites, or characters, within the program and identifying the *Event* that caused specific actions to be completed by the Sprites. Finally, students *Explore* making changes to code and describe how these changes affect program execution. This process provides critical support during the *Use* stage so that students can move on to making modifications and ultimately engage in creative exploration. Providing structured approaches such as Use→Modify→Create and TIPP & SEE to inquiry-based learning creates opportunities for success for diverse learners by equipping them with the programmatic logic necessary to engage in creative exploration without getting lost in the unlimited possibilities afforded by open-ended programming environments.

To engage students in disciplinary practices, we also integrated computational thinking skills with literacy development by aligning the curriculum with the California Standards for English Language Development and the Common Core State Standards for English Language Arts (see Table 2). There is evidence to support the relationship between learning to read, write, and program (Bers, 2019; Peppler & Warschauer, 2011). Through the interconnected aspects of coding and reading, students have multiple opportunities to examine the relationship between symbols and their corresponding meanings. For example, the Scratch program is embedded with multimodal symbols that can be arranged in logical ways to develop programs. Connections such as these provide engaging environments for children to develop their emergent literacy skills (Peppler & Warschauer, 2011).

Table 2
Sample Learning Goals That Integrate Grade 4 Common Core ELA, English Language Development, and Computer Science Teachers Association Standards (Jacob, Parker, & Warschauer, 2022)

Activity: Students program a story about their lives, families, or communities		
Computer science concepts: Loops, sequences, conditionals		
Computer Science Teachers Association (CSTA) Standards		
CSTA 1B-AP-10	Create programs that include sequences, events, loops, and conditionals	
CSTA 1B-AP-13	Use an iterative process to plan the development of a program by including others' perspectives and considering user preferences	
CSTA 1B-AP-15	Test and debug a program or algorithm to ensure it runs as intended	
English Language Development (ELD) Standards		
Emerging	Expanding	Bridging
3. Offering opinions Negotiate with or persuade others in conversations using basic learned phrases (e.g., I think) as well as open responses in order to gain and/or hold the floor.	3. Offering opinions Negotiate with or persuade others in conversations using a variety of learned phrases (e.g., That's a good idea. However …) as well as open responses, in order to gain and/or hold the floor.	3. Offering opinions Negotiate with or persuade others in conversations using a variety of learned phrases (e.g., That's a good idea. However …) as well as open responses in order to gain and/or hold the floor, elaborate on an idea, and provide different opinions.
11. Supporting opinions Offer opinions and provide good reasons (e.g., My favorite book is X because X) referring to the text or to relevant background knowledge.	11. Supporting opinions Offer opinions and provide good reasons and some textual evidence or relevant background knowledge (e.g., paraphrased examples from text or knowledge of content).	11. Supporting opinions Offer opinions and provide good reasons with detailed textual evidence or relevant background knowledge (e.g., specific examples from text or knowledge of content).
Corresponding English Language Arts Standards		
CCSS.ELA-L.SL.4.1 CCSS.ELA-L.SL.4.4 CCSS.ELA-L.SL.4.6 CCSS.ELA-L.W.4.9	**Engage** effectively in a range of collaborative discussions with diverse partners, building on others' ideas and expressing their own clearly. **Report** on a topic or text, tell a story, or recount an experience in an organized manner, using appropriate facts and relevant, descriptive details to support main ideas or themes; speak clearly at an understandable pace. **Differentiate** between contexts that call for formal English (e.g., presenting ideas) and situations where informal discourse is appropriate (e.g., small-group discussion); use formal English when appropriate to task and situation. **Draw** evidence from literary or informational texts to support analysis, reflection, and research.	

Through unplugged activities, Elementary Computing for All **builds on students' existing resources**. These unplugged activities leverage students' multiple meaning-making repertoires to support CS learning. For example, if children learn about loops, or a repeat block, first they stand up and dance, then they identify each move of the dance, then they identify how sequences of moves repeat, and then how the entire dance repeats. It is not until students fully embody the concept of loops that the term is presented. In this way, they use their everyday sense-making abilities to make sense of abstract concepts. Furthermore, the curriculum integrated culturally sustaining materials such as children's storybooks depicting diverse pioneers in the field of CS as well as memorable mentor videos highlighting the personal and professional journeys of Latine computer scientists and engineers.

Third, Elementary Computing for All **encourages rich classroom discourse** by providing multiple opportunities for collaboration such as pair programming, peer feedback opportunities, and collaborative projects. Furthermore, teachers were provided professional development on interactional patterns that encourage students to ask for clarification, use their everyday sense-making abilities to understand abstract concepts, and infer meaning from context.

Fourth, physical activities, unplugged activities, and visualizations were incorporated to **engage students in multiple modalities**, including but not limited to gestures, pictures, graphs, and symbols, which have been shown to simultaneously develop students' computational thinking and language skills (cf. Lee et al., 2019). For instance, through an unplugged exercise including movement and visual cues, students learn about key computational concepts such as parallelism, which refers to two actions happening at the same time. To learn about algorithms, or a list of steps that execute a function, students play Simon Says, in which the teacher instructs the class that they are a robot, and the students must provide instructions for the robot to move to a specific place in the classroom.

Fifth, by using sentence frames that teachers make available for use by students through student reflections, peer evaluation, feedback, and help-seeking, the curriculum provides **explicit teaching on how language functions in the discipline**. For each of the lessons, students were provided with placemats that contained the corresponding sentence frames.

Table 3
Computer Science Language Functions (Jacob, Parker, & Warschauer, 2022)

Teacher activities	*Student discourse*			*CS concepts (language function)*
	Emerging	*Expanding*	*Bridging*	
Remind students to think about the events that will cause each action to happen in their project, which programs will run parallel to each other, and how their project will reset once it has finished running.	I need help with __. __ caused ___ to happen. ___ and ___ are running at the same time. I used ___ to reset the program.	I am having difficulty with ___. __ is the event that caused __ to happen. __ and __ are running parallel to each other. I used _ to initialize the program.	Could you help me fix the following challenge in my code ___? The event that caused ___ to happen is ___. _ and _ are running parallel to each other/simultaneously/at the same time. __ caused the program to initialize.	Debugging, events, initialization, parallelism (Describing, comparing)

Instructional and Curricular Outcomes

Research on Elementary Computing for All has demonstrated that grounding the curriculum in effective practices for engaging multilingual students in STEM has resulted in positive social and academic outcomes for these students—and that elementary school teachers of such students are able to successfully teach the curriculum without any prior coding experience. For example, the curriculum helps predominantly low-income Latine students, including large numbers of students designated as English learners, learn the discourse of CS, advance their coding skills, and develop strong CS identities, all while achieving greater growth in their standardized reading and math skills than their peers who do not have access to the curriculum.

Student computational thinking outcomes

Findings from a cross-case study showed that students who received structured approaches to inquiry-based learning when receiving the Elementary Computing for All curriculum developed stronger computational thinking skills and CS identities (Jacob, Nguyen, et al., 2020). Furthermore, through the opportunities for self-expression embedded in the Scratch programming environment, students practiced key computational thinking practices such as (1) experimenting and iterating, (2) reusing and remixing, (3) testing and debugging, and (4) abstracting and modularizing (Jacob et al., 2020).

Developing students' CS identities

In a mixed-methods study, Jacob, Montoya, et al. (2022) found that students who participated in Elementary Computing for All significantly increased their CS identity attainment with respect to students' experiences with CS, their perceptions of CS, their perceptions of themselves as computer scientists, and their family support for CS. Follow-up interviews suggested that the connections students make between informal learning environments, rooted in students' homes, families, and communities, served to bolster their perceptions of their ability to do CS in the classroom. Building from this study, Jacob, Montoya, and Warschauer (2022) sought to understand the intersectional factors that shaped young Latinas' CS identity development. Findings indicated that parental support, cultural connections, and out-of-school learning helped these girls redefine stereotypes about who does CS and leverage the rich assets embedded in their diverse group membership.

CS disciplinary language learning

In a discourse analysis of peer-to-peer talk in students' video reflections of their Scratch projects, Nguyen et al. (2020) found that linguistic scaffolding enriched students' CS discourse, vocabulary usage, and description of computational thinking practices. Furthermore, students who elaborated on their projects and used more CS vocabulary had higher programming scores. This underscores the importance of peer-to-peer talk in strengthening connections between CS language usage and the development of computing skills. In a similar study on teacher noticing of students' discourse, findings indicated that at the beginning of the year, teachers honed in on students' CS vocabulary and concept learning. After several design meetings with Elementary Computing for All researchers, teachers noticed how students leveraged their everyday sense-making abilities to understand abstract CS concepts. Prior research suggests that paying attention to students' everyday language usage and their disciplinary language usage facilitates students' sense-making of complex and abstract phenomena and problems in STEM and CS (Gomez, 2007).

Teacher instructional outcomes

Teachers used instructional practices grounded in NASEM principles to engage students in integrated computational thinking, language, and literacy instruction. For example, they used strategies for activating prior knowledge, asking multiple questions to engage students in disciplinary practices; strategies for providing direct instruction that engaged students in multiple registers and modalities; and strategies for fostering student interaction and providing language support (Jacob, Parker, & Warschauer, 2022). They also used children's storybooks to leverage students' existing literacy skills to teach computational thinking. For example, through a read-aloud of "The Most Magnificent Thing," a story about a young girl who builds a computational artifact with her dog, one classroom teacher taught computational thinking concepts such as sequence, abstraction, debugging, and design through the NASEM strategies mentioned above (Jacob, Parker, & Warschauer, 2022).

Conclusion

Multilingual students are one of the most understudied and underserved groups in CS education. Furthermore, there is little data about students designated as ELs in CS courses. This exploratory research helps to ameliorate these issues by identifying practices and instructional material that are empirically effective in engaging multilingual students in CS and STEM.

Given the landscape surrounding CS education for students designated as ELs, it is critical for educators and practitioners to better understand the factors that shape CS learning for multilingual students and lead to more equitable CS instruction. This poses the question of how to move forward. There is much relevant work on providing effective practices for engaging multilingual students in STEM, but none of it is precisely targeted to the issue of how to successfully engage multilingual students in CS education. In this chapter, we draw on existing research, as well as our own and others' theoretical and empirical work to date, to put forth both a framework and curriculum for teaching CS to multilingual students.

Authors

Sharin Rawhiya Jacob is a Resesrcher in Computational Thinking Pathways at Digital Promise Global. Her research interests examine the linguistic and sociocultural factors that contribute to multilingual students' success in computing. Sharin was recently awarded the UC Irvine Public Impact Distinguished Fellowship for her commitment to bringing actionable change for multilingual students in computing.

Mark Warschauer is a Professor of Education and Informatics at the University of California, Irvine, and Director of the university's Digital Learning Lab. He is one of the most widely cited scholars on digital learning topics such as computer-assisted language learning, digital literacy, the digital divide, one-to-one laptop classrooms, and artificial intelligence in education.

Acknowledgments

We would like to thank the National Science Foundation (Grants no. 1738825 and no. 1923136) and the US Department of Education (Grant no. U411C190092) 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 or the US Department of Education.

References

Allen-Handy, A., Ifill, V., Schaar, R. Y., Rogers, M., & Woodard, M. (2020). Black girls STEAMing through dance: Inspiring STEAM literacies, STEAM identities, and positive self-concept. *Challenges and opportunities for transforming from STEM to STEAM education,* 198-219. IGI Global.

Ball, D., Tofel-Grehl, C., & Searle, K. A. (2017). Sustaining making in the era of accountability: STEM integration using e-textiles materials in a high school physics class. *Proceedings of the 7th Annual Conference on Creativity and Fabrication in Education*, 1-7. ACM. https://doi.org/10.1145/3141798.3141801

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

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.

Beyer, S. (2014). Why are women underrepresented in computer science? Gender differences in stereotypes, self-efficacy, values, and interests and predictors of future computer science course-taking and grades. *Computer Science Education, 24*(2-3), 153-192.

Bransford, J. D., Stipek, D. J., Vye, N. J., Gomez, L. M., & Lam, D. (2009). *The role of research in educational improvement*. Harvard Education Press.

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. *Proceedings of the 2012 Annual Meeting of the American Educational Research Association, Vancouver, Canada*, Vol. 1, 25.

Brown, B. A., & Ryoo, K. (2008). Teaching science as a language: A "content-first" approach to science teaching. *Journal of Research in Science Teaching*, *45*(5), 529-553.

Bybee, R. W. (1997). *Achieving scientific literacy: From purposes to practices*. Heinemann.

Cheryan, S., Plaut, V. C., Davies, P. G., & Steele, C. M. (2009). Ambient belonging: How stereotypical cues impact gender participation in computer science. *Journal of Personality and Social Psychology, 97*(6), 1045.

Coburn, C. E., Penuel, W. R., & Geil, K. E. (2013). *Practice partnerships: A strategy for leveraging research for educational improvement in school districts*. William T. Grant Foundation.

Code.org, CSTA, & ECEP Alliance. (2020). *2020 state of computer science education: Illuminating disparities*. https://advocacy.code.org/stateofcs

DiSalvo, B., Yardi, S., Guzdial, M., McKlin, T., Meadows, C., Perry, K., & Bruckman, A. (2011). African American men constructing computing identity. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 2967-2970). https://doi.org/10.1145/1978942.1979381

diSessa, A. A. (2000). *Changing minds: computers, learning, and literacy*. MIT Press.

Eisenhart, M., & Towne, L. (2003). Contestation and change in national policy on "scientifically based" education research. *Educational Researcher*, *32*(7), 31-38.

Erete, S., Martin, C. K., & Pinkard, N. (2017). Digital Youth Divas: A program model for increasing knowledge, confidence, and perceptions of fit in STEM amongst black and brown middle school girls. Y. rankin & J. Thomas (Eds.), *Moving students of color from consumers to producers of technology*, 152-173. IGI Global.

Estrella, G., Au, J., Jaeggi, S. M., & Collins, P. (2018). Is inquiry science instruction effective for English language learners? A meta-analytic review. *AERA Open*, *4*(2). https://doi.org/10.1177/2332858418767402

Franklin, D., Coenraad, M., Palmer, J., Eatinger, D., Zipp, A., Anaya, M., White, M., Pham, H., Gokdemir, O., & Weintrop, D. (2020). An analysis of use-modify-create pedagogical approach's success in balancing structure and student agency. *Proceedings of the 2020 ACM Conference on International Computing Education Research*, 14-24. https://doi.org/10.1145/3372782.3406256

García, O., & Wei, L. (2014). *Translanguaging: Language, bilingualism and education*. Palgrave Macmillan.

Gibbons, P. (2007). Mediating academic language learning through classroom discourse. In J. Cummins & C. Davison (Eds.), *International handbook of English language teaching*, 701-718. Springer.

Giere, R. N. (1999). *Science without laws*. University of Chicago Press.

Gomez, K. (2007). Negotiating discourses: Sixth-grade students' use of multiple science discourses during a science fair presentation. *Linguistics and Education*, *18*(1), 41-64.

Goode, J. (2008). Increasing diversity in K-12 CS: Strategies from the field. *Proceedings of the 39th SIGCSE technical symposium on computer science education*, 362-366. https://doi.org/10.1145/1352135.1352259

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*, 45-65. John Hopkins University Press.

Grapin, S. E., Llosa, L., Haas, A., & Lee, O. (2021). Affordances of computational models for English learners in science instruction: Conceptual foundation and initial inquiry. *Journal of Science Education and Technology*, 1-16.

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

International Society for Technology in Education (ISTE) & Computer Science Teachers Association (CSTA). (2011). *Operational definition of computational thinking for K-12 education*. http://www.iste.org/docs/ct-documents/computational-thinking-operational-definition-flyer.pdf

Irwin, V., Zhang, J., Wang, X., Hein, S., Wang, K., Roberts, A., York, C., Barmer, A., Mann, F. B., Dilig, R., Parker, S., Nachazel, T., Barnett, M., & Purcell, S. (2021). *Report on the condition of education 2021* (NCES 2021-144). National Center for Education Statistics.

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.

Jacob, S., Garcia, L., & Warschauer, M. (2020). Leveraging multilingual identities in computer science education. M. R. Freiermuth & N. Zarrinabadi (Eds.), *Technology and the psychology of second language learners and users*. Palgrave-Macmillan.

Jacob, S., Nguyen, H., Garcia, L., Richardson, D., & Warschauer, M. (2020). Teaching computational thinking to multilingual students through inquiry-based learning. *2020 Research on Equity and Sustained Participation in Engineering, Computing, and Technology (RESPECT)*, Vol. 1, 1-8. IEEE.

Jacob, S., Nguyen, H., Tofel-Grehl, C., Richardson, D., & Warschauer, M. (2018). Teaching computational thinking to English learners. *NYS TESOL Journal*, *5*(2), 12-24.

Jacob, S. R., Montoya, J., Nguyen, H., Richardson, D., & Warschauer, M. (2022). Examining the what, why, and how of multilingual student identity development in computer science. *ACM Transactions on Computing Education (TOCE)*, *22*(3), 1-33. https://doi.org/10.1145/3500918

Jacob, S. R., Montoya, J., & Warschauer, M. (2022). Exploring the intersectional development of computer science identities in young Latinas. *Teachers College Record, 124*(5), 166 –185.

Jacob, S. R., Parker, M. C., & Warschauer, M. (2022). Integration of computational thinking into English language arts. In A. Ottenbreit-Leftwich, A. Yadav (Eds.), *Computational thinking in preK-5: Empirical evidence for integration and future directions*, 55-63. ACM.

Jacob, S. R., & Warschauer, M. (2018). Computational thinking and literacy. *Journal of CS Integration*, *1*(1), 1-19.

Jadallah, M., Anderson, R. C., Nguyen-Jahiel, K., Miller, B. W., Kim, I.-H., Kuo, L.-J., Dong, T., & Wu, X. (2011). Influence of a teacher's scaffolding moves during child-led small-group discussions. *American Educational Research Journal*, *48*(1), 194-230.

Janzen, J. (2008). Teaching English language learners in the content areas. *Review of Educational Research*, *78*(4), 1010-1038.

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

Kelleher, C., & Pausch, R. (2007). Using storytelling to motivate programming. *Communications of the ACM, 50*(7), 58-64.

Kelly, G. J., & Chen, C. (1999). The sound of music: Constructing science as sociocultural practices through oral and written discourse. *Journal of Research in Science Teaching: The Official Journal of the National Association for Research in Science Teaching*, *36*(8), 883-915.

Kirschner, P. A., Sweller, J., & Clark, R. E. (2006). Why minimal guidance during instruction does not work: An analysis of the failure of constructivist, discovery, problem-based, experiential, and inquiry-based teaching. *Educational Psychologist, 41*(2).

Ladner, R. E., Seim, C., Sharif, A., Rizvi, N., & Glasser, A. (2021). Experiences of computing students with disabilities. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*, 939-940.

Latour, B. (1987). *Science in action: How to follow scientists and engineers through society*. Harvard University Press.

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

Lee, O., Llosa, L., Grapin, S., Haas, A., & Goggins, M. (2019). Science and language integration with English learners: A conceptual framework guiding instructional materials development. *Science Education*, *103*(2), 317-337.

Lemke, J. L. (1990). *Talking science: Language, learning, and values*. Ablex.

Manz, S. L. (2002). A strategy for previewing textbooks: Teaching readers to become THIEVES. *The Reading Teacher, 55*(5), 434.

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

McFarland, J., Hussar, B., Zhang, J., Wang, X., Wang, K., Hein, S., ... & Barmer, A. (2019). The Condition of Education 2019. NCES 2019-144. National Center for Education Statistics.

McGinn, M. K., & Roth, W. M. (1999). Preparing students for competent scientific practice: Implications of recent research in science and technology studies. *Educational Researcher*, *28*(3), 14-24.

Michaels, S., & O'Connor, C. (2015). Conceptualizing talk moves as tools: Professional development approaches for academically productive discussion. In *Socializing intelligence through talk and dialogue*, 347-362.

Mishra, S., Balan, S., Iyer, S., & Murthy, S. (2014). Effect of a 2-week Scratch intervention in CS1 on learners with varying prior knowledge. *Proceedings of the 2014 Conference on Innovation & Technology in Computer Science Education*, 45-50. ACM.

National Academies of Sciences, Engineering, and Medicine. (2018). *English learners in STEM subjects: Transforming classrooms, schools, and lives*. https://www.nap.edu/catalog/25182/english-learners-in-stem-subjects-transforming-classrooms-schools-and-lives

National Center for Science and Engineering Statistics. (2021). *Women, minorities, and persons with disabilities in science and engineering: 2021*. https://ncses.nsf.gov/pubs/nsf21321/report/field-of-degree-minorities

National Research Council. (1996). *National science education standards*. National Academy Press. (ERIC Document Reproduction Service No. ED 391 690)

Nguyen, H., Garcia, L., Jacob, S., Richardson, D., & Warschauer, M. (2020). Teachers' use of video reflections to reinforce computer science language and concepts. In *2020 Research on Equity and Sustained Participation in Engineering, Computing, and Technology (RESPECT)*, Vol. 1, 1-8. IEEE.

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

Peppler, K. A., & Warschauer, M. (2011). Uncovering literacies, disrupting stereotypes: Examining the (dis)abilities of a child learning to computer program and read. *International Journal of Learning and Media, 3*(3), 15-41.

Prado, Y., Jacob, S., & Warschauer, M. (2022). Teaching computational thinking to exceptional learners: Lessons from two inclusive classrooms. *Computer Science Education, 32*(2), 188-212. https://doi.org/10.1080/08993408.2021.1914459

Reiser, B. J. (2004). Scaffolding complex learning: The mechanisms of structuring and problematizing student work. *The Journal of the Learning Sciences*, *13*(3), 273-304.

Rosebery, A. S., Warren, B., & Conant, F. R. (1992). Appropriating scientific discourse: Findings from language minority classrooms. *The Journal of the Learning Sciences*, *2*(1), 61-94.

Royal, D., & Swift, A. (2016, October 18). *U.S. minority students less exposed to computer science*. Gallup. http://www.gallup.com/poll/196307/minority-studentsless-exposed-computer-science.aspx

Ryoo, K., & Bedell, K. (2019). Supporting linguistically diverse students' science learning with dynamic visualizations through discourse-rich practices. *Journal of Research in Science Teaching*, *56*(3), 270-301.

Salac, J., Thomas, C., Butler, C., Sanchez, A., & Franklin, D. (2020). TIPP&SEE: A learning strategy to guide students through use-modify Scratch activities. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, 79-85.

Sengupta, P., Dickes, A., & Farris, A. (2018). Toward a phenomenology of computational thinking in STEM education. *Computational Thinking in the STEM Disciplines*, 49-72.

Shea, L. M., & Shanahan, T. B. (2011). Methods and strategies: Talk strategies. *Science and Children, 49*(3), 62-66.

Snow, M. A., Met, M., & Genesee, F. (1989). A conceptual framework for the integration of language and content in second/foreign language instruction. *TESOL Quarterly*, *23*(2), 201-217.

Swanson, L. H., Bianchini, J. A., & Lee, J. S. (2014). Engaging in argument and communicating information: A case study of English language learners and their science teacher in an urban high school. *Journal of Research in Science Teaching*, *51*(1), 31-64.

Sweller, J., & Cooper, G. A. (1985). The use of worked examples as a substitute for problem solving in learning algebra. *Cognition and Instruction, 2*, 59-89.

Sykes, E. R. (2007). Determining the effectiveness of the 3D Alice programming environment at the CS I level. *Journal of Educational Computing Research, 36*(2), 223-244. https://doi.org/10.2190/J175-Q735-1345-270M

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., Hoadley, C., Castillo, A. R., & Ascenzi-Moreno, L. (2020). Languages, literacies and literate programming: Can we use the latest theories on how bilingual people learn to help us teach computational literacies? *Computer Science Education*, 1-24.

Vogel, S., Santo, R., & Ching, D. (2017). 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*, 609-614. https://doi.org/10.1145/3017680.3017755

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.

Wilensky, U., Brady, C. E., & Horn, M. S. (2014). Fostering computational literacy in science classrooms. *Communications of the ACM*, *57*(8), 24-28. https://dl.acm.org/doi/pdf/10.1145/2633031

Windschitl, M. (2003). Inquiry projects in science teacher education: What can investigative experiences reveal about teacher thinking and eventual classroom practice? *Science Education*, *87*(1), 112-143.

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

Zhou, N., Cao, Y., Jacob, S., & Richardson, D. (2020). Teacher perceptions of equity in high school CS classrooms. *ACM Transactions on Computing Education (TOCE)*, *20*(3), 1-27.

Zwiep, S. G., & Straits, W. J. (2013). Inquiry science: The gateway to English language proficiency. *Journal of Science Teacher Education*, *24*(8), 1315-1331