JCSI Journal of Computer Science Integration

Volume 1 | Issue 1 Article 1

2018

Computational Thinking and Literacy

Sharin Rawhiya Jacob
University of California, Irvine

Mark Warschauer
University of California, Irvine

Recommended Citation

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

JCSI
Journal of Computer Science Integration

Computational Thinking and Literacy

Abstract

Today's students will enter a workforce that is powerfully shaped by computing. To be successful in a changing economy, students must learn to think algorithmically and computationally, to solve problems with varying levels of abstraction. These computational thinking skills have become so integrated into social function as to represent fundamental literacies. However, computer science has not been widely taught in K-12 schools. Efforts to create computer science standards and frameworks have yet to make their way into mandated course requirements. Despite a plethora of research on digital literacies, research on the role of computational thinking in the literature is sparse. This conceptual paper proposes a three dimensional framework for exploring the relationship between computational thinking and literacy through: 1) situating computational thinking in the literature *as* a literacy; 2) outlining mechanisms by which students' existing literacy skills can be leveraged to foster computational thinking; and 3) elaborating ways in which computational thinking skills facilitate literacy development.

Keywords

computational thinking, literacy, computational literacy, computer science, K-12

DOI

10.26716/jcsi.2018.01.1.1

Corresponding Author

Sharin Rawhiya Jacob, University of California, Irvine, Irvine, CA 92697, USA.
Email: sharinj@uci.edu

Today's students will enter a workforce that is powerfully shaped by computing. To be successful in a changing economy, students must learn to think algorithmically and computationally to solve problems with varying levels of abstraction. These computational thinking skills have become so integrated into social function as to represent fundamental literacies. However, computer science has not been widely taught in K-12 schools. Efforts to create computer science standards and frameworks have yet to make their way into mandated course requirements. Despite a plethora of research on digital literacies, research on the role of computational thinking in the literature is sparse.

One purpose of this article is to define computational thinking as a new form of literacy by integrating well-known literature on computational literacy, new literacy studies, new media studies, and computer literacy. Specific social, cognitive, and material features serve to distinguish this new form from other types of literacy. A second purpose is to examine how students' existing literacy skills facilitate computational thinking and vice versa. Well-researched efforts to integrate computer programming into K-12 literacy instruction illustrate these relationships.

The article opens by describing computational thinking and moves on to provide a conceptual definition of literacy that draws from well-known sociocultural perspectives. We then propose a threefold theoretical framework for exploring the relationship between computational thinking and literacy. The framework begins with situating computational thinking in the educational literature as a literacy in and of itself. The next section explores the mechanisms and pedagogical devices by which students' existing literacy skills can be leveraged to foster computational thinking. Finally, this perspective is inverted, in order to consider ways in which computational thinking skills facilitate traditional and new literacies development. Given the paucity of research on the intersection between computing and literacy development, a conceptual survey such as this is necessary in order to uncover salient issues for further examination and research.

Definition and status of computational thinking in K-12 education

In 2006, Jeanette Wing published an influential piece in the *Journal of the Association for Computing Machinery* titled "Computational Thinking." Her claim that computational thinking skills are generalizable across disciplines initiated a groundbreaking discussion on the role of computer science in solving pressing problems across essential domains of the human experience. Computer science is defined as the study of computers, including their hardware, software, algorithmic processes, applications, and impact on society (Tucker et al., 2006). In contrast, computational thinking is a generalized problem-solving approach applicable to a wide array of STEM and non-STEM fields. A formal definition is still an open topic of discussion in the literature (Barr & Stephenson, 2011; Grover, & Pea, 2013), but overall, scholars agree that computational thinking skills include algorithmic thinking, navigating multiple levels of abstraction, decomposing problems into manageable pieces, and representing data through models. Computational thinking can be taught with or without the use of computers (Bell, Alexander, Freeman, & Grimley, 2009), but it is often operationalized through computer programming.[1] Abstraction lies at the heart of computational thinking (Bennedsen & Caspersen,

[1] Code literacy describes the ability to program, and ultimately refers to the teaching and learning of reading and writing in computer programming languages (diSessa, 2000; Hockly, 2012; Prensky, 2008; Rushkoff, 2012; Vee, 2013). Computational thinking integrates code literacy as a means of operationalizing inherent computational skills,

2006; Kramer, 2007), including the abilities to identify patterns, find the underlying principles governing these patterns, and generalize from first principles.

Efforts in the United States to train students in computational thinking are largely unrealized, due to the lack of a systematically mandated computer science curriculum. The US lags behind other nations' efforts,[2] despite acknowledgment by governmental and other entities that providing high-quality computer science instruction to all students in the US presents a major opportunity for national and local education reform. The Computer Science for All initiative, initiated by President Obama in 2016, seeks to equip K-12 students with computational thinking skills that drive technological innovation. In a review of the literature, Buitrago Flórez et al. (2017) highlighted efforts of startup companies and computer science associations, including the Association for Computing Machinery, Google, and Microsoft, to promote computer science research through the Computing at the Core coalition. The Association for Computing Machinery, Code.org, the Computer Science Teachers Association, the Cyber Innovation Center, and the National Math and Science Initiative collaboratively developed the K-12 Computer Science Framework that will guide the development of state and district-wide standards and curriculum.

Conceptual framework for defining literacy

Educational policy on literacy focuses primarily on cognitive and psychological approaches (Muth & Perry, 2010; Pearson & Hiebert, 2010) that rely on development of traditional skills, such as decoding, fluency, and comprehension (Perry, 2012). Sociocultural approaches, on the other hand, emphasize the social, economic, cultural, and political circumstances that give rise to literary practices. Vygotsky (1978) defined communication as emerging from interactions between children and adults in specific environments, dispelling psychological views of learning as solely representing internal mental processes. A significant body of research on literacy practices draws from sociocultural perspectives (Barton & Hamilton, 1998; Gee, 2000).

Approaching literacy from a functional linguistics perspective, Halliday (1973) maintained that language is inseparable from social context: Language is an embodiment of culture. Gee (1996) further proposed that language instantiates not only culture, but also social interactions, power, politics, and values. Warschauer (1999) examined literacy from a historical perspective, identifying the role of technology in literacy development, particularly how innovations in technology result in evolving definitions of literacy.[3] This article draws on these and other sociocultural and material approaches to define literacy as a set of practices situated in a sociocultural context (Barton & Hamilton, 1998; Gee, 1996) that utilizes external technological media to enable expression. To the extent that literacy practices consist of internal mental representations, we examine how these representations are themselves governed by prevailing sociocultural values.

---

which are essential to coding as well as to other disciplines in both programming and non-programming environments.

[2] Both Israel and Germany mandate computer science in high school. Countries including Russia, South Africa, New Zealand, and Australia have taken steps to integrate CS into the K-12 curriculum. The Royal Society of the United Kingdom recently chartered policy efforts to introduce computing to all school children.

[3] From the printing press of the mid-15th century, to the industrial revolution, to the advent of the computer, technology plays an integral role in defining what constitutes 'literacy' and the requisite skills that are valued in each of these historical paradigms.

DOI: 10.26716/jcsi.2018.01.1.1

Three-dimensional framework for examining computational thinking and literacy

We now turn to a theoretical discussion on the relationship between computational thinking and literacy. Figure 1 illustrates a three-dimensional framework that 1) represents computational thinking *as* a new literacy embedded in modern sociocultural practices (computational thinking *as* literacy); 2) discusses how literacy development can be leveraged to foster computational thinking (computational thinking *through* literacy); and 3) explores ways in which computational thinking can facilitate literacy development (literacy *through* computational thinking).

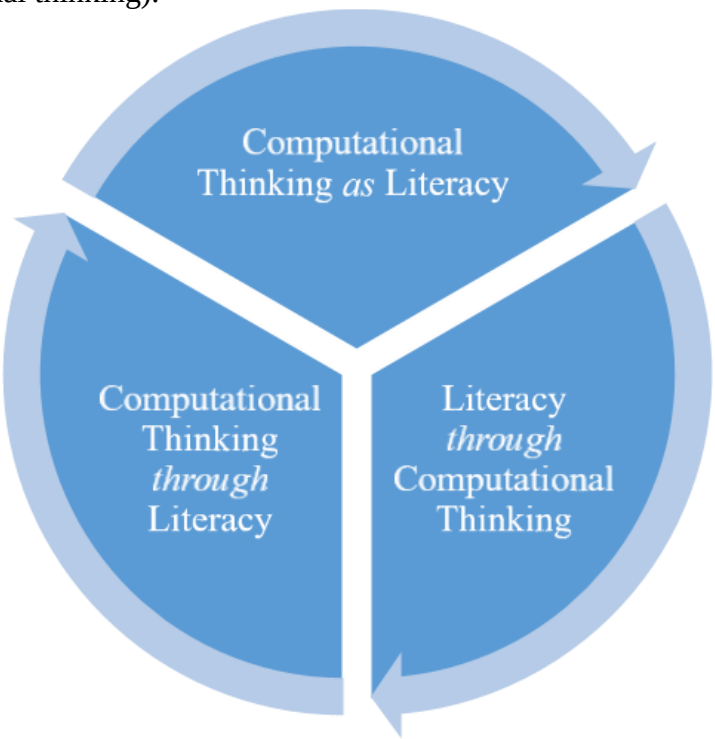

Figure 1. A three-dimensional framework for understanding computational literacy

Computational thinking *as* literacy

Wing's (2006) vision of computational thinking revitalizes diSessa's (2000) view of computational literacy, leading researchers sometimes to use the terms interchangeably (Grover & Pea, 2013). Both diSessa and Wing promote computational competence as a means for transforming social and cognitive practices through computing. DiSessa outlines a theoretical framework for defining computational literacy that takes into account the material, cognitive, and social dimensions of computing.[4] Wing similarly acknowledges the role of the computer as a medium for bringing about social and cognitive transformation, cognitive approaches to problem solving that govern computational processes, and social milieus that give rise to computational

[4] The cognitive dimension deals with benefits that external representational systems (e.g., text, graphing, algebra, and computing systems) bestow on individuals. The material view explores how external mediums, such as computers, enable expression and the development of new capabilities. The social view explores the conditions under which computational competence takes hold in society.

thinking.

Computational literacy: transformation of literacy practices through material supports. Papert (1980) and diSessa (2000) have proposed a goal of universal computational literacy, arguing that it should be taught to all students as a mechanism for providing access to 'powerful ideas.' In addition to suggesting that all students should learn to code, diSessa argues that increasing computational competence in the population will transform culture in the same way that the printing press transformed literacy in the 15th century. Programming utilizes computers as a "meta-medium," or material support, enabling human expression to evolve in new and unpredictable ways. From clay, papyrus, and wax to the printing press, computers, and the internet, these technologies have transformed the way in which communication takes place in time and space and have subsequently led to new conceptions of literacy (Warschauer, 1999).

In framing his vision of universal computational literacy, diSessa (2000) sought to define literacy itself as: "a socially widespread patterned deployment of skills and capabilities in a context of material support … to achieve valued intellectual ends" (p. 19). Since then, literacy's definition has expanded beyond the symbolic and rhetorical capabilities necessary for meaningful communication to include the tools, material supports, or technologies that make such communication possible (diSessa, 2000; Forte & Guzdial, 2004; Vee, 2013; Warschauer, 1999).[5] Modern-day programming provides support for traditional written communication, such as word processing, as well as instantaneous communication such as tweets, Facebook posts, texts, and instant messages (Vee, 2013).[6] As new communication technologies become available, material supports have the potential to become fundamental, or 'infrastructural' (Vee, 2013) to existing social practices. In fact, they often become so fundamental that their utility is rendered essentially invisible.

Despite these innovations in communication technologies, prevailing social and cultural ideologies value certain literary media over others. Cultural reluctance to adopt digital media is evident in reports such as the NEA's *Reading at Risk*, which claims that print represents an "irreplaceable" medium for fostering "intellectual capability" while digital media foster "shorter attention spans and accelerated gratification" (Bradshaw & Nichols, 2004, pp. vii-viii). Furthermore, K-12 schools, with their emphasis on state-mandated reading and math assessments, tend to value traditional literacy development and thus print-based mediums over multimedia modes of instruction. This lag in adopting contemporary notions of literacy prevents institutions from updating pedagogical practices to reflect paradigmatic shifts in social, cultural, and economic conditions.

Computational literacy: the role of cognition in achieving intellectually valued ends. Computational literacy facilitates new ways of thinking to achieve valued intellectual ends (diSessa, 2000). The human intellect works cooperatively with material supports to achieve these ends. For example, diSessa (2000) underscores the benefits algebraic notation afforded to individuals in comparison to print literacies by typing Galileo's theories of motion in words alongside the corresponding algebraic notation. The juxtaposition highlights the ease with which

[5] They take on specific forms, including inscription systems such as languages, and various sub-literacies such as algebraic notation and programming languages. These forms are disseminated by specific designs, or tools (e.g., pen and paper, print, and computers).

[6] These material supports allow mental representations and forms to become "reproducible, manipulatable, transportable" (diSessa, 2000, p. 6).

DOI: 10.26716/jcsi.2018.01.1.1

algebraic representations result in increased understanding compared to traditional alphabetic print. It also provides a convenient analogy for how the cooperative relationship between computation and the human intellect more efficiently carry out thought processes. The cognitive skills and emergent practices that programmers use in developing new media (diSessa, 2000; Livingstone, 2004) involve abstraction, automation, and analysis (Lee et al., 2011).

Abstraction requires decomposition, or stripping problems to their most essential pieces, and it is often attained through pattern generalization, or finding the underlying principles that govern patterns in a problem or model (Bennedsen & Caspersen, 2006; Kramer, 2007). For example, to simulate the epidemiology of infectious diseases in a given location, one needs to identify the aspects of the location (e.g., doorways, windows, ventilation) that contribute to the spread of disease, while ignoring other aspects that do not contribute to contagion. Abstraction relies chiefly on the human intellect to set parameters for a specific model or problem.

Automation refers to the process of instructing a computer to execute a series of tasks, or algorithms, quickly and efficiently (Lee et al., 2011). When running the same simulation, automated random number generators can create travel vectors for the pathogen, without requiring a researcher to program each individual movement. Loops are frequently utilized to automate such movement, for both random and patterned motions. While the setting of parameters relies chiefly on the human intellect, automation assigns the heavy lifting of repeated computations to computers, which possess greater processing power than humans. Computer programs are essentially automations of abstractions (Lee et al., 2011; National Research Council, 2010; Wing, 2011). Computational thinking facilitates and extends the human intellect to achieve socially and culturally valued ends, such as eliminating the spread of contagion.

Analysis involves a reflective determination of whether abstractions and automations are correct, that is, whether external representations are accurate and sufficient to embody challenging problems (Lee et al., 2011). Analysis represents interaction between the human intellect and material supports (i.e., programming languages) that push the boundaries of understanding in addressing problems whose solutions were once considered unachievable. The bulk of contemporary mathematics and engineering utilize computational methods to identify solutions that evade the human intellect, such as curing cancer and eliminating hunger. In some cases of human–computer interaction, for example in artificial intelligence, the computer plays the role of learner. To this end, the human intellect facilitates and transforms computational processes.

Computational literacy: the ubiquitous nature of computation in social practices. Defining computational thinking as a literacy provides a way to unify well-researched theories on literacy instruction. According to diSessa (2000), the decision to define a particular practice as a literacy relies heavily on socially constructed contexts. Thus, defining computational thinking as such necessitates due consideration. A literacy is operationalized through the patterned and widespread deployment of a material intelligence (diSessa, 2000). The variations in forms and patterns of a given type of literacy represent literary genres (romance novels, science fiction) that serve specific purposes in specific contexts (pleasure reading). Likewise, the various genres of programming languages (agent-based, object-oriented, visual) enable a host of modern social communications such as email, texting, and word processing (Vee, 2013). Furthermore, whether a given genre is commonly adopted relies heavily on social, cultural, economic, and historical conditions. As social progress and advancement are inextricably linked with technological innovation, given the rise of automation, its utilization is increasing at

astonishing rates. In particular, previous research has demonstrated that computing and new media have become inextricably linked to the relationships, identities, and recreational activities of children and youth (Buckingham, 2000; Livingstone & Bovill, 2001; Valentine, Skelton, & Chambers, 1998).

New literacies, new media studies, and computing. Following diSessa's (2000) definition, studies of new literacies situate literacy practices within sociocultural contexts (Barton, 1991; Bruce, 2002; Gee, 1996; Moje, 2000; Street, 2003). New media studies constitute a parallel literature encompassing literacies that extend beyond traditional print-based media (Gee, 2010). New media provide several affordances to literacy acquisition by presenting students with increasingly complex textual representations. Requiring more than the ability to read and write, new media promote the utilization of multiple modalities to express ideas, such as technological fluency, media literacy, visual literacy, aural literacy, and critical literacy (Peppler & Warschauer, 2011). Furthermore, navigating new media provides semiotically rich resources in which textual genres intersect to create new hybrid forms and conventions (Bruce, 1997; Mills, 2010). These features of new media have been examined primarily in relation to the consumption of digital media, such as surfing the web, playing videogames, blogging, and using social media. Research on computational thinking and computer programming in new media studies, which is unique in engaging students in creating these applications and platforms for broader consumption, is sparse.

As the definition of literacy evolved to encompass new socially and culturally constructed forms, new media studies shifted the emphasis from contemporary media criticism to media production (Gee, 2010). Media production includes using preprogrammed digital software (such as iMovie) to remix, repurpose, and create new digital productions. Although students creatively produce new media in this model, they remain consumers of technology due to the utilization of prepackaged platforms as conduits of self-expression. Lynch (2015) observed that computation and algorithmic processes inherently make up the software that is used in educational spaces, and these myriad technologies (i.e., e-readers, blogs, hypertext) are produced by human beings and companies who adhere to philosophical frames that lend themselves to computation and coding. The decisions that creators of these software make, as well as the nature of the software itself, affect pedagogy in subtle ways, providing both affordances and limitations. Engaging students in computational thinking and programming, on the other hand, empowers students to become developers and creators of these digital platforms, thereby transforming literacy practices by shaping the pedagogical capabilities and constraints of the software itself, arguably encompassing new media *production* in a more essential form.

The integration of computer literacies and new media studies through computational thinking. Computer literacy and new media studies, previously thought of as distinct fields, both incorporate computational thinking as a form of new media production. Computer literacy refers to the ability to understand and use computer hardware and software and was initially distinguished from information literacy, or accessing data for problem solving and evaluation (Horton Jr., 1983).[7] Notions of computer literacy have for the most part evolved independently of new media and literacy studies. Kafai and Peppler (2011) synthesized new

[7] The emergence of information and computer technology (ICT) served to blur this distinction. Definitions of computer literacy evolved to include the ability to generate information and to express oneself creatively using computer-based media (National Research Council, 1999).

media studies with literature on computer literacy, arguing that creative design and production are at the core of both research agendas, even if scholars of computer literacy more strongly emphasized the technical aspects of programming and hardware engineering. While creative media design and production represent a significant link between the two fields, little research positions computational thinking as an essential literacy in itself.

Defining computational thinking as a literacy integrates the literatures on computational literacy, new literacies studies, new media studies, and computer literacy. Consider Gee's (1996) definition of literacy as a set of practices situated in a sociocultural context that utilize external technological mediums to enable expression. Computational thinking represents cognitive skills, habits, and approaches to solving problems that are relevant across multiple domains of the human experience. Furthermore, the computer is the external medium that enables the reproduction, manipulation, and transportation of literary genres while computation facilitates the human intellect to carry out processes more efficiently. The skills and practices emerging from computational thinking, such as the abilities to work with multiple levels of abstraction, carry out processes more efficiently through automation, and test and analyze solutions, provide beneficial affordances to individuals that achieve socially and culturally valued ends. Finally, computational thinking shares elements with creative media use, as media design and production represent an iterative process that requires programmatic logic at each stage.

Computational thinking *through* literacy

This section examines well-researched efforts to integrate computational thinking into traditional literacy instruction. This integration offers several benefits. For example, rather than waiting for top-down curriculum reform, teachers can bring computer science directly to current instructional practices, such as to English Language Arts and English as a Second Language classrooms (see Jacob et al., 2018). Correspondingly, incorporating computational thinking and programming into authentic literacy practices will better align literacy instruction with the needs of a changing economy. As students utilize problem-solving skills as a vehicle for self-expression, they are also motivated to become capable citizens in an information-driven society.

Algorithmic thinking involves decoding textual or block-based programming commands and sequencing them in syntactically and semantically meaningful ways. Pane and Myers (2001) examined how non-programming fifth graders express solutions in natural language to common programming problems, and found that, for this purpose, natural language utterances lack precision as compared to textual programming commands. A pedagogical technique for fostering computational competence entails engaging students in using 'pseudocode' to write algorithms that can then be translated into actual code. Pseudocode is an effective method for presenting introductory programming concepts such as conditionals, loops, Boolean expressions, and variables to novice programmers (Malan & Leitner, 2007). In writing pseudocode, students of all ages utilize their existing language skills to develop computational competence; they write algorithms with varying degrees of precision and iteratively revise them to more closely reflect computational syntax. In this way, they leverage their natural language abilities to understand algorithmic thinking and practice textual programming languages. For example, teachers can ask middle and high school students to write directions for a simple card game using pseudocode (i.e., if/then statements) and then guide them in revising their code to more closely match a text-based language.

There has been much discussion about whether narrative structures can serve as a

heuristic to capture the semiotic processes that underlie computational thinking. Utilizing linguistic, semiotic, and cognitive perspectives, de Souza and colleagues (2011) analyzed the interpretive processes involved in programming for middle school students participating in agent-based game creation. De Souza et al. compared students' verbal analysis of game structure to their execution of game design, with a particular focus on transitive verbs. For example, a student description "the hunter killed the monkey" is actually encoded in the game as "the monkey disappears when it touches the hunter." Or in the case of diffusion, tokens erase themselves, such as when a frog disappears or 'erases itself' when it encounters water. To program games, students used their existing knowledge of transitive verb structures to set parameters for the game and, as they became more familiar with the game's algorithmic processes, began to describe the game using intransitive verbal structures that more closely represented the programming language. This reveals the ability to develop and adapt language structures to more closely match the precision of gaming algorithms. Although students who generated complex verbal narratives did not necessarily exhibit advanced programming execution, the study highlighted semiotic associations between narrative descriptions and encoding of games that point toward the use of narrative structures to scaffold computational thinking skills development. In practice, teachers can ask students who are programming games to describe their design process (i.e., the problem or obstacle, the characters in the game, the gaming system, the parts of their game, the rules of their game, how the points are scored, etc.) and revisit and revise their descriptions at different points throughout the programming process.

One method for integrating computational thinking into the curriculum involves giving students programming software to construct narratives. Programming languages such as Scratch, and Storytelling Alice offer storytelling tutorials that provide emerging coders access to computational methods (Burke, O'Byrne, & Kafai, 2016). Both Scratch and Alice improve students' confidence in their programming abilities (Cooper, Dann, & Pausch, 2000; Kalelioglu, & Gülbahar, 2014). Presenting a low floor and high ceiling (Harvey & Mönig, 2010), Scratch and Alice are meant to serve as entry points into programming, with Scratch typically taught in elementary and middle school and Alice in middle and entry-level high school courses, and sometimes in introductory university courses such as CS1. Research indicates Scratch and Alice are effective in transitioning students into text-based languages such as C and C++. For example, findings from a 2-week Scratch intervention in a CS1 course on novice programmers showed advanced programmers using C++ reached 80% of the complexity of highly rated novice Scratch programs and 69% of students perceived Scratch to be useful in transitioning to C++ (Mishra, Balan, Iyer, & Murthy, 2014). Findings from a quasi-experimental design using repeated measures found that students in a CS1 course using Alice performed better on performance tasks than the comparison group of students using C (Sykes, 2007).

Burke and Kafai (2012) examined narratives from middle school participants in a writer's workshop for youth programmers. They found that storytelling in Scratch offered several benefits, including developing traditional literacy skills (narrative and composition skills), fostering new literacies, increasing technological fluency, and introducing coding to students at an early age. Furthermore, students' existing knowledge of the writing process (drafting, revising, editing) supported their understanding of computational processes (design, troubleshooting, debugging). An analysis of digital artifacts found that students were adept at utilizing programming concepts such as coordination and synchronization, parallel execution, loops and event handling, but that more advanced programming concepts such as Boolean logic, conditional statements, and variables were largely missing. This finding is widely corroborated

in the literature (Burke & Kafai, 2010; Grover, Pea, & Cooper, 2016; Maloney, Peppler, Kafai, Resnick, & Rusk, 2008).

Although a shortcoming of programming digital stories is arguably the decreased use of the most advanced programming concepts, the creative and relatively unrestricted medium of self-expression often motivates students who would not otherwise view themselves as computer scientists (Kelleher & Pausch, 2007; Peppler & Kafai, 2007). Kelleher and Pausch (2007) found that girls engaged in Storytelling Alice were three times more likely to practice programming on their own and more inclined to enroll in a future course that utilized the program. Motivating factors included increased opportunities for self-expression, drawing on personal experience, and sharing stories with classmates. Peppler and Kafai (2007) corroborate the motivating effects of media-rich programming environments such as Scratch, which are achieved by tapping into prior knowledge to facilitate self-expression. Furthermore, the medium reinforces constructivist approaches to learning which are inherently more engaging for students. For example, Clayton and Ardito (2009) found that constructivist activities utilizing new technologies allow students multiple opportunities to develop a sense of authorship and ownership of their learning.

Computational fluency can be further integrated into instruction by promoting identity investment through programming multimodal media. Cummins, Hu, Markus, and Montero (2015) explored the impact of identity development on academic and literacy achievement through 'identity texts.' These texts engage low socioeconomic status, multilingual, and marginalized groups in leveraging their multimodal and multilingual skills to create literature and art that reflect their personal and social identities. As a result, students construct identities as multiliterate, academically engaged citizens, thereby challenging societal power structures that devalue their cultures. As students use programming software to construct identity texts, they develop computational thinking skills while maximizing cognitive engagement, identity investment, community interaction, and language use and development (Cummins, Hu, Markus, & Montero, 2015). In practice, 'identity texts' such as dual-language books that portray children and their families as protagonists and draw from their histories, lives, and interests can be created in virtual programming environments such as Scratch and Alice.

Peppler and Kafai (2007) point to convergence culture as a means of engaging students in culturally relevant participation. Convergence culture represents the merger of old and new media (Jenkins, 2006) and "emphasizes how one can choose among many forms to participate socially in new media cultures" (Peppler & Kafai, 2007, p. 4). Convergence culture connotes a participatory, as opposed to receptive, culture that values speaking, writing, and producing. At the Computer Clubhouse, an after-school programming club that serves predominately African-American and Latino youth, students who engaged in computational thinking through creative media production were better equipped with the knowledge, skills, and attitudes necessary for participating in the rapidly changing digital landscape (Peppler & Kafai, 2007). Whereas underrepresented youth were once constrained to critical analysis of prepackaged interfaces, through creative media design students are able to insert their voices into the dominant culture as active participants. Creating and producing new media in settings such as the Clubhouse has the potential to broaden access and participation in both new media culture and the field of computer science.

Literacy *through* computational thinking

Teaching *literacy through computational thinking* involves leveraging existing

computational thinking and coding skills to promote language and literacy development. For students with varying skill levels, programming can serve as a doorway into traditional and new literacies development. Peppler and Warschauer (2011) examined the development of emergent literacies through the creative use of technology in a case study on Brandy, a child with intellectual disabilities who participated in an after-school computer clubhouse. Brandy was able to leverage her programming skills in Scratch to develop emergent literacies such as metalinguistic awareness, reading and writing skills, self-efficacy, and a sense of identity as a computer scientist (Peppler & Warschauer, 2011). This study and other findings call for further exploration of the affordances computer programming poses for emergent literacy development.

Contrastive analysis of the similarities and differences between programming and natural languages has the potential to benefit competent programmers who are developing their language skills. Bers, Flannery, Kazakoff, and Sullivan (2014) paraphrase Stair and Reynolds (2003) to define a program as a "sequence of instructions that a computer…acts out in an order specified by the programmer" (p. 150). While the sequence must be arranged using proper syntax, each command has a specific meaning, the sequencing of which results in actions being taken by the computer. Bers and colleagues (2014) pointed to correspondence as encapsulating an understanding of how each command represents a specific action taken by the computer. During computer science instruction, these distinctions among form (programming syntax), meaning (command semantics), and use (correspondence between commands and actions) can be leveraged to facilitate language development. This is precisely because a parallel distinction can be mapped onto language instruction, conceived as a dynamic process wherein students attend to the form, meaning, and use of syntactic structures, thus incorporating linguistic areas of semantics and pragmatics into instruction (as cited in Larsen-Freeman & Long, 2014, p. 264). Framing programming and natural languages in this way is particularly beneficial to speakers of other languages who possess computational competence but may lack linguistic proficiency.

Language approaches that focus on form, for example, rely heavily on sequencing of syntactic structures. Sequencing involves placing objects and ideas in the correct order (Zelazo, Carter, Reznick, & Frye, 1997) and is a fundamental component of algorithmic thinking (Pea & Kurland, 1984). Elementary and middle school lessons on sequencing textual and block-based commands can be integrated into structural language lessons that rely on sequencing as a heuristic. This skill can help students navigate the complexities of grammatical forms, as evidenced by myriad elaborate constructions, such as the ordering of sentence-final adverbials, *wh*-question inversion (or lack thereof), and the ordering of determiners in noun phrases (Celce-Murcia & Larsen-Freeman, 1999). Sequencing can also facilitate writing at the suprasentential level, such as organizing paragraphs into topic sentences and main ideas, correctly ordering events in a story, or writing instructions for carrying out a specific task.

Just as students cannot master programming syntax without understanding the semantic meaning of commands, they cannot produce correct linguistic forms without considering their corresponding meanings. In fact, block-based programming environments such as Scratch are aimed at developing students' programmatic logic before their syntactical knowledge (Malan & Leitner, 2007). The process of deciphering meaning from programming syntax can be extrapolated to grammar acquisition, particularly to the development of semantic knowledge construction. To return to the ordering of sentence-final adverbials, for example, notions of manner, direction, time, frequency, and purpose are categorically semantic in nature. Thus, students must be able to distinguish whether the constructions (e.g., *slowly, to the bank, once a week,* and *because he has to study for the test*) answer the right questions (e.g., *how, where, how*

often, and *why*) before attempting to sequence them correctly. Lessons that move beyond basic decoding and encoding of programming languages to elicit semantic representations of commands, such as the use of command cards to program bots, can frontload lessons that require semantic knowledge to master challenging linguistic constructs.

Programming is not limited to learning syntax and semantics. Likewise, the three-pronged approach (form, meaning, and use) underscores the idea that language lessons designed exclusively around a series of linguistic structures and semantic representations do not result in knowledge transfer, an arguable shortcoming of skills-based teaching approaches. To explain, as focus on form implies learning a set of rules, knowledge *about* language, or declarative knowledge does not necessarily result in procedural knowledge, or the automatic *use* of language in communicative contexts. Grover (2015) highlights "algorithmic flow of control: sequence, looping constructs, and conditional logic" as concepts foundational to computational competence (p. 3). It is often through debugging, or assessing the degree to which a string of code achieves desired results, that students develop metacognitive understanding of programming (Papert, 1980). If procedural knowledge is the goal, learning to code requires doing code and learning from one's mistakes. Traditional language is similarly learned through socially mediated communication, in which students learn from and assess their errors as they attempt to express meaning in contextually appropriate ways. Collaborative programming activities such as pair programming thematic design projects can engage students in the types of social negotiation that lead to pragmatic discourse about computational and literacy topics.

Beyond the sentential and suprasentential levels, programming has been shown to reinforce genre-based writing development. Unlike the bulk of digital storytelling research, which examines how existing narrative writing skills facilitate programming, Burke and Kafai (2010) observe the process conversely, focusing on how coding can reinforce storytelling and composition skills. For example, middle school students participating in an after-school club learned about character development through programming stories in Scratch. While certain characters required unique code to generate considerable variability in behavior, other characters relied on loops to replicate a narrow set of behaviors, thus reinforcing the distinction between main and supporting character development. Furthermore, as students learned to switch backdrops, they learned about not only setting, but also the necessity to stage scenes successively in order to build to a climax and resolution.

Conclusion

Given its many applications, the definition of "literacy" in computer science often depends on the domain to which computational skills are applied. Early researchers viewed "computational literacy" as a set of programming skills students could apply to computational concepts in mathematics. Informed by applications in other disciplines, computational literacy has since expanded to consider the cognitive, material, and social dimensions of computing. A parallel literature exploring the role of digital media in new literacies identifies literacy as a set of practices embedded in a sociocultural context (Barton & Hamilton, 1998; Gee, 1996). By focusing on design and presentation of digital media, scholars of new literacies emphasize creative expression through preprogrammed digital platforms over the programming skills that generate new media technologies. Synthesizing these disparate approaches from the fields of computational literacy, new literacy studies, and computer literacy, provides a new foundational perspective on the interactions between computational thinking and literacy.

We hold that computational thinking is a new literacy, with a programmatic logic that drives new media production. Future research should focus on gaining a better understanding of the material, cognitive, social, and creative processes involved in the learning of computational thinking. Insights gleaned from this research can be used to improve pedagogical practices and deepen our understanding of computational competence as it relates to current K-12 literacy practices.

This approach opens new questions to investigate. The nascent literature on assessments of computational thinking warrants further exploration of the learning outcomes emphasized through this integration. Effectively measuring these outcomes would provide insight into how computational thinking contributes to emergent literacy development, such as in Brandy's case study (Peppler & Warschauer, 2011). It would also help to uncover how learning programmatic logic contributes to traditional and new literacies development. Though this work focuses narrowly on integrating computational thinking with narrative writing genres, analyzing how writing in other genres fosters computational thinking would provide new insights to better align literacy objectives with computer science instruction.

This work introduces an overarching heuristic for comparing programming and natural languages. Building on this heuristic, a detailed contrastive analysis into these similarities and differences would not only enhance the teaching of programming languages, but also of traditional language teaching for those students who already possess computational competence. Findings from Portnoff (2018) indicate that introductory programming courses activate natural language processing areas of the brain, as opposed to math or logic. If learning programming is similar to language learning, then utilizing knowledge from second language acquisition research to teach programming could provide affordances for both computer science and traditional language instruction. These synergies are readily apparent not only in the form, meaning, and use heuristic provided, but also for example, when comparing objects, methods, and properties in the various programming languages to nouns, verbs, and adjectives. Exploring further similarities would enhance both the learning of languages through computer programming and vice versa.

This work also provides a stepping stone to explore how the creative use of new media contributes to issues of participation and equity in computer science. In addition to limited conceptions of literacy development, problem solving in K-12 schools, particularly in math and science, has been taught within predefined contexts that allow for narrow sets of conclusions. Computer science curriculum, on the other hand, engenders creative and innovative problem solving approaches that lead to many potential solutions. Mistakes are often viewed as opportunities for debugging programs, which in turn reinforce learning. Insofar as these classroom practices engender diversity of ideas and provide multiple pathways to mastery, they are conducive to promoting equity and cultural responsivity. They are also conducive to equitable language teaching in which students are given multiple opportunities to participate in social and academic discourse communities. Research on equitable computer science teaching practices, and how they translate into equitable literacy teaching practices has the potential to broaden participation in computer science while enhancing these integrative instructional practices.

Implications

The integration of computer science with literacy practices will also increase students' exposure to computer science. The complex and ubiquitous nature of computer science

complicates efforts to create equitable computer science classrooms. Computer science's integration into K-12 curriculum is relatively new. Yet, with its numerous applications, computer science permeates our daily life and society much more than traditional disciplines such as math and science (Grover & Pea, 2013). To this end, students who are equipped with the skills to be successful computational thinkers are more likely to be students with more social resources and access to new technologies, while underserved students will typically lack exposure to these technologies in their home environment and social life (Goode, 2008). Extending the teaching and learning of computational thinking to literacy instruction, and not just math and science, will increase the types and amount of exposure that these students receive, preparing them to be successful in the new economy. Furthermore, the manner in which virtual programming environments such as Scratch and Alice enable self-expression serves to diversify a curriculum that will broadly appeal to students who come from nontraditional backgrounds. Exploring how these educational programming platforms can connect to students' homes, cultures, and communities can further enhance their integration into current literacy practices.

Computational thinking is the critical skill of the 21st century. While educational policy initiatives have focused on assessing literacy by measuring discrete reading and writing skills, the economy continues to value computational skills that solve pressing problems across a wide array of disciplines. Although computer science initiatives take time to affect policy, cross-curricular integration of computational thinking represents a mechanism for bringing computer science instruction directly to K-12 classrooms. In particular, integrating computational thinking with current literacy practices leverages students' existing literacy skills to improve computational outcomes, and conversely, fosters students' literacy development through the practice of computing. To the degree that new literacies have been taught in K-12 schools, students have primarily been consumers of new technologies. Teaching computational thinking, beginning in elementary grades, will empower students to become developers and creators of new technologies. The focus will shift from 'what' students produce within prepackaged platforms to 'how' to formulate thoughts and express oneself in a manner that is accessible to a computer to achieve desired results. This shift will lead students to engage in the types of creativity and problem solving that drive production, advance knowledge, improve outcomes, and enable progress.

References

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

Barton, D. (1991). The social nature of writing. In *Writing in the community*, ed. D. Barton and R. Ivancic, 55–78. Newbury Park, CA: Sage.

Barton, D., & Hamilton, M. (1998). *Local literacies. Reading and writing in one community.* London/New York: Routledge.

Bell, T., Alexander, J., Freeman, I., & Grimley, M. (2009). Computer science unplugged: School students doing real computing without computers. *The New Zealand Journal of Applied Computing and Information Technology*, *13*(1), 20-29.

Bennedsen, J., & Caspersen, M. E. (2006). Abstraction ability as an indicator of success for learning object-oriented programming?. *ACM SIGCSE Bulletin*, *38*(2), 39-43. https://doi.org/10.1145/1138403.1138430

Bers, M. U., Flannery, L., Kazakoff, E. R., & Sullivan, A. (2014). Computational thinking and tinkering: Exploration of an early childhood robotics curriculum. *Computers & Education*, *72*, 145-157. https://doi.org/10.1016/j.compedu.2013.10.020

Bradshaw, T., & Nichols, B. (2004). Reading at risk: A survey of literary reading in America. research division report# 46. *National Endowment for the Arts*.

Bruce, B. C. (1997). Critical issues literacy technologies: What stance should we take? *Journal of Literacy Research*, *29*(2), 289-309. https://doi.org/10.1080/10862969709547959

Bruce, B. C. (2002). Diversity and critical social engagement: How changing social technologies enable new modes of literacy in changing circumstances. In *Adolescents and literacies in a digital world*, ed. D. E. Alvermann, 1–18. New York, NY: Peter Lang.

Buckingham, D. (2000). *After the death of childhood: Growing up in the age of electronic media*. Malden, MA: Polity Press.

Buitrago Flórez, F., Casallas, R., Hernández, M., Reyes, A., Restrepo, S., & Danies, G. (2017). Changing a Generation's Way of Thinking: Teaching Computational Thinking Through Programming. *Review of Educational Research*. *87*(4), 834-860. https://doi.org/10.3102/0034654317710096

Burke, Q., & Kafai, Y. B. (2010, June). Programming & storytelling: opportunities for learning about coding & composition. In *Proceedings of the 9th international conference on interaction design and children* (pp. 348-351). ACM. https://doi.org/10.1145/1810543.1810611

Burke, Q., & Kafai, Y. B. (2012, February). The writers' workshop for youth programmers: digital storytelling with scratch in middle school classrooms. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education* (pp. 433-438). ACM.

Burke, Q., O'Byrne, W. I., & Kafai, Y. B. (2016). Computational Participation. *Journal of Adolescent & Adult Literacy*, *59*(4), 371-375. https://doi.org/10.1002/jaal.496

Celce-Murcia, M., & Larsen-Freeman, D. (1999). *The Grammar Book* (2nd ed.). Boston, MA: Heinle ELT.

Clayton, C. D., & Ardito, G. (2009). Teaching for ownership in the middle school science classroom: Towards practical inquiry in an age of accountability. *Middle Grades Research Journal*, *4*(4), 53-79.

Cooper, S., Dann, W., & Pausch, R. (2000, April). Alice: A 3-D tool for introductory programming concepts. In Journal of Computing Sciences in Colleges (Vol. 15, No. 5, pp. 107-116). Consortium for Computing Sciences in Colleges.

Cummins, J., Hu, S., Markus, P., & Kristiina Montero, M. (2015). Identity texts and academic achievement: Connecting the dots in multilingual school contexts. *TESOL Quarterly*, *49*(3), 555-581. https://doi.org/10.1002/tesq.241

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

de Souza, C., Garcia, A., Slaviero, C., Pinto, H., & Repenning, A. (2011). Semiotic traces of computational thinking acquisition. *End-User Development*, 155-170. https://doi.org/10.1007/978-3-642-21530-8_13

Forte, A., & Guzdial, M. (2004, January). Computers for communication, not calculation: Media as a motivation and context for learning. In *System Sciences, 2004. Proceedings of the 37th Annual Hawaii International Conference on* (pp. 10-pp). IEEE.

Gee, J. P. (1996). *Social linguistics and literacies: Ideology in discourses* (2nd ed). London: Falmer.

Gee, J. P. (2000). New people in new worlds: Networks, the new capitalism and schools. *Multiliteracies: Literacy learning and the design of social futures*, *4368*.

Gee, J. P. (2010). *New digital media and learning as an emerging area and "worked examples" as one way forward*. Cambridge, MA: MIT Press.

Goode, J. (2008). Increasing diversity in k-12 computer science. *ACM SIGCSE Bulletin*, 40, 362. https://doi.org/10.1145/1352322.1352259

Grover, S. (2015, April). "Systems of Assessments" for Deeper Learning of Computational Thinking in K-12. In *Proceedings of the 2015 Annual Meeting of the American Educational Research Association* (pp. 15-20).

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. (2016, February). Factors influencing computer science learning in middle school. In *Proceedings of the 47th ACM technical symposium on computing science education* (pp. 552-557). ACM. https://doi.org/10.1145/2839509.2844564

Halliday, M. A. K. (1973). *Explorations in the functions of language*. London: Hodder.

Harvey, B., & Mönig, J. (2010). Bringing "no ceiling" to scratch: Can one language serve kids and computer scientists. In Proceedings: Constructionism, Paris, France.

Hockly, N. (2012). Digital literacies. *ELT journal*, *66*(1), 108-112. https://doi.org/10.1093/elt/ccr077

Horton Jr, F.W., 1983. Information literacy vs. computer literacy. *Bulletin of the American Society for Information Science*, *9*(4), 14-16.

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.

Jenkins, H. (2006). *Convergence culture: Where old and new media collide*. New York: NYU press.

Kafai, Y. B., & Peppler, K. A. (2011). Youth, technology, and DIY: Developing participatory competencies in creative media production. *Review of Research in Education*, *35*(1), 89-119. https://doi.org/10.3102/0091732X10383211

Kalelioglu, F., & Gülbahar, Y. (2014). The Effects of Teaching Programming via Scratch on Problem Solving Skills: A Discussion from Learners' Perspective. *Informatics in Education*, *13*(1).

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

Kramer, J. (2007). Is abstraction the key to computing?. *Communications of the ACM*, *50*(4), 36-42. https://doi.org/10.1145/1232743.1232745

Larsen-Freeman, D., & Long, M. H. (2014). *An introduction to second language acquisition research*. New York, NY: Routledge.

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

Livingstone, S. (2004). Media Literacy and the Challenge of New Information and Communication Technologies. *The Communication Review*, *7*, 3-14. https://doi.org/10.1080/10714420490280152

Livingstone, S., & Bovill, M. (2001). *Families and the internet: an observational study of children and young people's internet use.* Public report. London School of Economics and Political Science.

Lynch, T. L. (2015). Where the machine stops: Software as reader and the rise of new literatures. *Research in the Teaching of English*, *49*(3), 297.

Malan, D. J., & Leitner, H. H. (2007). Scratch for budding computer scientists. *ACM SIGCSE Bulletin*, *39*(1), 223-227. https://doi.org/10.1145/1227504.1227388

Maloney, J. H., Peppler, K., Kafai, Y., Resnick, M., & Rusk, N. (2008). Programming by choice: urban youth learning programming with scratch. *ACM*, *40*(1), 367-371. https://doi.org/10.1145/1352322.1352260

Mills, K. A. (2010). A review of the "digital turn" in the new literacy studies. *Review of Educational Research*, *80*(2), 246-271. https://doi.org/10.3102/0034654310364401

Mishra, S., Balan, S., Iyer, S., & Murthy, S. (2014, June). Effect of a 2-week scratch intervention in CS1 on learners with varying prior knowledge. In Proceedings of the 2014 conference on Innovation & technology in computer science education (pp. 45-50). ACM.

Moje, E. B. (2000). "To be part of the story": The literacy practices of gangsta adolescents. *Teachers College Record*, *102*(3), 651–90. https://doi.org/10.1111/0161-4681.00071

Muth, W. R., & Perry, K. H. (2010). Adult literacy: An inclusive framework. *Handbook of research on teaching the English Language Arts, Third Edition.* New York: Routledge.

National Research Council (1999). *Being fluent with information technology*. Washington, DC: National Academy Press.

National Research Council. (2010). *Report of a workshop on the scope and nature of computational thinking*. National Academies Press.

Pane, J. F., & Myers, B. A. (2001). Studying the language and structure in non-programmers' solutions to programming problems. *International Journal of Human-Computer Studies*, *54*(2), 237-264. https://doi.org/10.1006/ijhc.2000.0410

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

Pea, R. D., & Kurland, D. M. (1984). On the cognitive effects of learning computer programming. *New Ideas in Psychology*, *2*(2), 137-168. https://doi.org/10.1016/0732-118X(84)90018-7

Pearson, P. D., & Hiebert, E. H. (2010). National reports in literacy: Building a scientific base for practice and policy. *Educational Researcher*, *39*(4), 286-294. https://doi.org/10.3102/0013189X10370205

Peppler, K. A., & Kafai, Y. B. (2007). From SuperGoo to Scratch: Exploring creative digital media production in informal learning. *Learning, Media and Technology*, *32*(2), 149-166. https://doi.org/10.1080/17439880701343337

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.

Perry, K. H. (2012). What Is Literacy?--A Critical Overview of Sociocultural Perspectives. *Journal of Language and Literacy Education*, *8*(1), 50-71.

Portnoff, S. R. (2018). The introductory computer programming course is first and foremost a language course. *ACM Inroads*, *9*(2), 34-52. https://doi.org/10.1145/3152433

Prensky, M. (2008). The role of technology. *Educational Technology*, *48*(6).

Rushkoff, D. (2012). Code literacy: A 21st-century requirement. *Edutopia*, *2016*.

Stair, R. M., & Reynolds, G. W. (2003). Principles of Information Systems, Thomson Learning.

Street, B. (2003). What's "new" in new literacy studies? Critical approaches to literacy in theory and practice. *Current Issues in Comparative Education* 5 (2):77–91. http://www.tc.edu/cice/Issues/05.02/52street.pdf (accessed January 12, 2012).

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

Tucker, D., McCowan, F., Deek, C., Stephenson, J., Jones, J., & Verno, A. (2006). A model curriculum for k-12 computer science: Report of the acm k-12 task force computer

science curriculum committee. Technical report, Association for Computing Machinery, New York, NY.

Valentine, G., Skelton, T., & Chambers, D. (1998). Cool places: An introduction to youth and youth cultures. *Cool places: Geographies of youth cultures*, 1-32.

Vee, A. (2013). Understanding computer programming as a literacy. *Literacy in Composition Studies*, *1*(2), 42-64. https://doi.org/10.21623/1.1.2.4

Vygotsky, L. (1978). Interaction between learning and development. *Readings on the Development of Children*, *23*(3), 34-41.

Warschauer, M. (1999). Electronic literacies: Language, culture and power in online education. New York, NY: Routledge.

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

Wing, J. M. (2011). Proceedings from 2011 Frontiers in Education Conference IEEE: *Computational thinking*. https://doi.org/10.1109/VLHCC.2011.6070404

Zelazo, P. D., Carter, A., Reznick, J. S., & Frye, D. (1997). Early development of executive function: A problem-solving framework. *Review of General Psychology*, *1*(2), 198. https://doi.org/10.1037/1089-2680.1.2.198