The Code-Centric Nature of Computational Thinking Education: A Review of Trends and Issues
in Computational Thinking Education Research

Vance Kite, Soonhye Park Ph.D., and Eric Wiebe Ph.D.

All authors are located in the North Carolina State University College of Education: Department
of STEM Education

North Carolina State University
Department of STEM Education
208 Poe Hall
2310 Stinson Dr.
Raleigh, NC 27695

vkite@ncsu.edu
spark26@ncsu.edu
wiebe@ncsu.edu

All correspondence for this article should be directed to Vance Kite (vkite@ncu.edu)

Abstract

Computational thinking (CT) is being recognized as a critical component of student success in the digital era. Many contend that integrating CT into core curricula is the surest method for providing all students with access to CT. However, the CT community lacks an agreed-upon conceptualization of CT that would facilitate this integration, and little effort has been made to critically analyze and synthesize research on CT/content integration (CTCI). Conflicting CT conceptualizations and little understanding of evidence-based strategies for CTCI could result in significant barriers to increasing students' access to CT. To address these concerns, we analyzed 80 studies on CT education, focusing on both the CT conceptualizations guiding current CT education research and evidence-based strategies for CTCI. Our review highlights the code-centric nature of CT education and reveals significant gaps in our understanding of CTCI and CT professional development for teachers. Based on these findings we propose an approach to operationalizing CT that promotes students' participation in CT, present promising methods for infusing content with CT, and discuss future directions for CT education research.

Keywords: Computational thinking, Programming, Integrating computational thinking, Computational thinking education, Unplugged

1. Introduction

The landscape of work is fundamentally shifting as the ever-growing knowledge economy increasingly requires a coupling of human cognitive and computing power. Computation is reshaping the aims, materials, and methods mathematicians and scientists employ in the generation of knowledge, and has been harnessed by fields as different as textile manufacturing is from economics (Orton et al., 2016). However, it has been widely reported that there is a shortage of students prepared to enter such a technologically-focused workforce (Montoya, 2017). In recognition of this challenge, computational thinking (CT) has been identified as a fundamental component of readying students for this rapidly-changing world of work (Grover & Pea, 2018; Heintz, Mannila, & Färnqvist, 2016; The Royal Society, 2012).

Seymour Papert (1980) first described using computer programming to teach disciplinary content in his seminal work *Mindstorms: Children, Computers, and Powerful Ideas*. Papert (1980) asserted that computers should be tools for creation and stimulators of metacognition; rather than “a means of putting children through their paces” (p. 19). Many of the skills that Papert believed students developed while working in his LOGO programming environment would later be recognized as the core practices of the software engineering community. In 2006, Jeannette Wing labeled these practices computational thinking (CT) and argued that they are as important as basic literacy and mathematical proficiency. Importantly, she theorized CT as being distinct from the field of computer science (CS), especially in that CT entails conceptualizing rather than programming, fundamental skills rather than rote syntax skills, and human thought based in creativity, not programmed computer-thought (Wing, 2006). Thus, CT has come to be understood as a set of practices that requires both knowledge and skills, rather than a static body of knowledge (Weintrop et al., 2016). In the decade following Wing’s (2006) article, computer scientists, education researchers, and teachers’ organizations worked to define the practices under the CT umbrella (e.g., Barr & Stephenson, 2011; Brennan & Resnick, 2012; Computer Science Teachers Association, 2011; Wing, 2008, 2010). Though the CT education community has not settled on a consensus framework for CT, researchers are beginning to focus on pedagogical approaches to teaching CT practices, such as, pattern recognition, problem decomposition, abstraction creation, algorithm development, and debugging (Grover & Pea, 2018; Shute, Sun, & Asbell-Clarke 2017).

The lack of consensus on key CT practices, however, is problematic (Moreno-León, Román-González, & Robles, 2018; Shute et al., 2017). Kuhn (1962) explains that the commonly-held epistemologies, theories, and concepts of a discipline dictate the type of questions that are asked and tools that are employed in the pursuit of knowledge. Similarly, Bettis & Gregson (2001) argue that “a scientific paradigm can be thought of as an all-encompassing way of thinking that organizes scientific endeavors; it is a pair of glasses in which we “see” the world” (p. 3). Thus, disparate conceptualizations of CT could result in fractured and contentious operationalizations of the discipline and interpretations of research results; complicating efforts to bring efficacious CT practices to K-12 classrooms (Hestness, Ketelhut, McGinnis, and Plane, 2018). Thus, it is imperative to understand the nature of the CT conceptualizations that have guided the CT education research community to illuminate the potential implications of competing CT conceptualizations for CT research and education.

Despite the reported lack of a consensus conceptualization of CT, most agree on the critical importance of CT for all students. Notable CT scholars argue that integrating CT into core curriculum is the most promising strategy for providing a diverse group of students with broad access to these critical tools (e.g., Grover & Pea, 2018; Jona et al., 2014; Repenning et al., 2015;

Sengupta, Kinnebrew, Basu, Biswas, & Clark., 2013). Globally, Europe has seen rapid growth in the number of CT tools and approaches utilized for CT integration into elementary grade classrooms, thus exposing teachers and students to CT practices in the early grades (Bers, 2018). In Taiwan, Li and colleagues (2016) have developed a code-free activity for teaching decomposition that could be integrated into a range of high school STEM courses. An approach used in Israel has been the inclusion of CS/CT items on national exams as a top-down strategy for inclusion of CT into STEM curricula (Zur-Bargury, Parv, & Lanzberg, 2013). Similarly, U.S. science education has taken a significant step towards CT integration through identifying CT as one of eight essential science practices to be taught through the recently-released Next Generation Science Standards (NGSS Lead States, 2013). The National Science Foundation in the U.S. has also released a number of grant opportunities that support research on CT education.

While early CT frameworks (e.g. Brennan & Resnick, 2012; CSTA, 2011) emphasized programming skills and concepts, the NGSS has taken a different approach by emphasizing an integrative and cross-disciplinary approach that moves beyond programming (NGSS Lead States, 2013). Building upon the NGSS notion that CT is more than just writing computer programs and that CT should be integrated into the core curricula, Weintrop et al. (2016) set forth the Computational Thinking in Mathematics and Science Taxonomy (CT-MS) (Table 1.); which outlines 22, broadly-applicable skills that can be integrated into mathematics and science curriculum. Weintrop and colleagues (2016) note, and we underscore, that “the practices are highly interrelated and dependent on one another” (p. 134).

Table 1

The Computational Thinking in Mathematics and Science Taxonomy (CT-MS) (Weintrop et al., 2016).

CT-MS Category	CT-MS Practices
Modeling and Simulation	Using computational models to understand a concept Using computational models to find and test solutions Assessing computational models Designing computational models Constructing computational models
Systems Thinking	Investigating complex systems as a whole Understanding the relationships within a system Thinking in levels Communicating information about a system Defining systems and managing complexity
Data Practices	Collecting data Creating data Manipulating data Analyzing data Visualizing data
Computational Problem Solving	Preparing problems for computational solutions Programming Choosing effective computational tools

Assessing different approaches/solutions to a problem
 Developing modular computational solutions
 Creating computational abstractions
 Troubleshooting and debugging

Given that recent reviews of CT literature highlight that CT encompasses a set of broadly-applicable problem-solving practices, CT should lend itself to integration in a variety of core curricula (Flórez et al., 2017; Haseski, Ula, & Tu., 2018; Grover & Pea, 2018; Moreno-León et al., 2017). Indeed, research has demonstrated that integrating CT instruction in required subject-area content enhances learning of both content and CT while exposing the broadest-possible range of students to CT practices (Jona et al., 2014; Orton et al. 2016; Repenning et al. 2015; Tatar, Harrison, Stewart, Frisina, & Musaeus 2017). Unfortunately, CT integration in core curricula is not yet common practice. Schute and colleagues (2017) attribute this problem to the lack of consensus about the framing of CT and the lack of research-supported best practices for CT integration; while Yadav, Hong, and Stephenson (2016) identify limited professional development support for teachers as a major obstacle for CT integration. Taken together, providing all students with the opportunity to develop their CT proficiency rests on the creation of effective professional development programs for teachers that focus on implementation of research-based CT integration strategies that are grounded in agreed-upon conceptualizations of CT.

Considering the issues regarding CT education research discussed above, we conducted a thematic review of CT education literature that was guided by the following research questions: (1) What is the nature of the CT conceptualizations and CT operationalizations that are guiding current CT education research? (2) What strategies have been proposed or tested for integrating CT into core curriculum? Findings from our work clarify the ways that researchers have conceptualized CT through their writing and operationalized CT through their study designs, revealing “blind spots” in the existing literature. In addition, our review of research-guided strategies for CT integration provides important insight into strategies for a computational pedagogy (CP), with important implications for both teacher professional development and CT instructional practices.

2. Method

2.1. Identifying and Screening Literature for Review

The process of literature identification, screening, and selection was conducted iteratively as we identified, refined, and verified emerging themes (Gough, Oliver, & Thomas, 2012). Literature for review was first searched, screened, and selected twice in 2017, then again in fall of 2019. To constrain our initial search to the field of education, we conducted a keyword search in the ERIC database using the term “computational thinking” anywhere in a journal article published between 2006 and 2017 and constrained the results to peer-reviewed journal articles. We began our search in 2006 because that is the publication date for Wing’s (2006) article (*Computational Thinking*) which catalyzed interest in CT education. As a result, 78 articles were located and the abstracts of all articles were screened using the following eligibility criteria:

1. Empirical studies of CT education initiatives.
2. Conducted with K-12 grade students (5-18 years of age) or K-12 teachers.
3. In either formal or informal educational settings.

Articles such as Grover and Pea (2013) and Weintrop et al. (2016) were captured in the search but excluded from our analysis because they do not present the results of an empirical study. Likewise, Yuen and Robbins (2015) was captured but excluded because the study was conducted with undergraduate students. Ultimately, 20 articles from the ERIC database were included in our first in-depth analysis.

Next, to capture manuscripts not located in the ERIC database, we queried Google Scholar in the summer of 2017 using the search term *allintitle*: “Computational Thinking” and constrained the search timeframe to articles published after 2006. 428 full-text articles appeared in this search. Since we aimed to critically analyze the literature by focusing on the depth of detail in which each research question is addressed (Gough & Thomas, 2012), we limited the number of studies included in our initial review to 40. Thus, we used a random number generator to select 20 additional articles (Pitol & De Groote, 2014). If a selected article was either a duplicate from the ERIC search or did not meet our selection criteria, we proceeded sequentially down the list of search results; reviewing the abstract of each article until an article was found that met our inclusion criteria. This process was repeated until an additional 20 eligible articles were located. These studies were analyzed in-depth to identify preliminary emerging themes.

To expand the breadth of the literature in our analysis and to refine the emerging themes, we conducted two additional rounds of Google Scholar searches in the fall of 2019. The first search followed the above-mentioned search, screening, and selection process, except that the timeframe spanned from 2006-2019. This search resulted in 2560 manuscripts, from which a random number generator was used to select 25. Finally, in an effort to include a robust sampling of the most recent literature, an additional search was conducted using the timeframe of 2017-2019. Fifteen articles were randomly selected from this set. In total, 80 articles are included in our analysis. This sample will be referred to as the *analysis sample*.

The selection process was performed by two of the three authors. The first author of this review compiled a list of studies, and then he and the second author screened them with the eligibility criteria while discussing discrepancies until they reached agreement. As presented in Appendix A, our analysis sample includes studies from twenty-six countries published in twenty-four unique conference proceedings, twenty-seven academic journals, and three books. Of the studies analyzed, the first was published in 2010, 13 studies (16%) were published between 2011 and 2014, 63 studies (79%) were published between 2015 and 2018, and three (4%) were published in 2019.

2.1.1. Selection of literature for analysis sample validation. Finally, a validation check was conducted to analyze the representativeness of the analysis sample of 80 articles against the larger corpus of scholarly CT literature. To focus on literature considered “high impact,” a two pronged approach was utilized. First, Web of Science (WoS) was queried using the identical criteria employed to obtain the 80 analysis sample articles. This curated database is among the most widely used by scholars in the sciences (Martín-Martín, Orduna-Malea, Thelwall, & López-Cózar, 2018). This search resulted in 654 records, of which one was excluded as a duplicate. To compliment the curated sources from WoS, a query of Google Scholar was done in late 2019 using the same criteria, resulting in 3095 records. Using title and citation information, the following steps were conducted: 1) The WoS and Google Scholar records were merged and duplicates removed, 2) records missing information (e.g., authors, place of publication, abstract) were removed, 3) records published in a non-English language were removed, 4) titles that did not match the inclusion criteria were removed. This resulted in 1181 records remaining. Hirsch’s (2005) h-index was then used to select Google Scholar records that represented the more highly cited work

not contained in the Web of Science corpus. This resulted in a final pool of 547 records for which full citations and abstracts were acquired.

This strategy was informed by prior work demonstrating the complementary nature of the data contained in Web of Science and Google Scholar (Halevi, Moed, & Bar-Ilan, 2017; Martin-Martin et al., 2018). Whereas WoS contains a narrower, curated collection, Google Scholar provides broad coverage, especially for fields not well covered in WoS, including Computer Science (Halevi et al., 2017). Since Google Scholar provided a broader, less curated collection, Hirsch's (2005) h-index was used to identify the most impactful work in this set; a method recognized as a high quality, complementary method to Clarivate Analytics' Web of Science impact factor calculation method (Harzing, 2009; Martin-Martin et al., 2018).

The combined Web of Science/Google Scholar corpus of citations and abstracts were reviewed for the inclusion criteria stated at the beginning of Section 2.1, resulting in 239 retained empirical, peer-reviewed studies. This corpus will henceforth be referred to as the *validation sample*. The analysis procedures for the validation sample are described in Section 2.2.1.

2.2. Analysis of the Selected Literature

Analytic process. Our review process represents a combination of a thematic review and a mapping review/systematic map (Grant & Booth, 2009). That is, this review intended to critically examine a diverse set of documents to “take stock” (Grant & Booth, 2009, p. 93) of the current state of CT education research, while synthesizing analysis results into a systematic map of the literature. The following section provides a detailed description of our analytic process relating to the first research question.

2.2.1. CT conceptualization and operationalization analysis. To understand the relationships between CT conceptualizations and study designs, each of the eighty studies was analyzed in terms of (1) CT conceptualizations, (2) study purpose, (3) instructional tools, (4) study design, (5) study context, and (6) measures utilized. We adopted the constant comparative method (Strauss & Corbin, 1990) for analysis steps one, two, and six. *A priori* codes (Creswell & Poth, 2018) derived from the National Research Council (NRC) (2002) were employed for analysis step four. The analysis results of the individual studies were then synthesized into the Computational Thinking Conception Design Map (CTCDM) presented in Figure 1.

Step 1: Computational Thinking Conceptualizations. By conceptualization, we refer to authors' choices regarding the CT scholars, frameworks, and terms highlighted in their writing. Two coders employed multiple rounds of open and axial coding (Corbin & Strauss, 2015) to identify commonalities in CT scholars cited, CT frameworks employed, and CT terms highlighted across the literature. As a result, twenty-four open codes emerged that were aggregated into four sub-categories, which were then aggregated through axial coding into two categories, *CT as Code-Centric Skills* and *CT as Interdisciplinary Practices (IDP)* (see Table 2 for an excerpt from our codebook).

Step 2: Study Purpose. The purpose of each study was analyzed through open coding, which resulted in sixteen codes. These sixteen codes were then grouped into larger categories through an axial coding process that constantly compared and contrasted the similarities and differences across the codes (Corbin & Strauss, 2015). This procedure produced six categories as shown in the Purpose column of Table 2.

Step 3: Instructional Tools. The studies were divided into two categories (Plugged or Unplugged) based on their use of technology. If an intervention used any form of electronic

technology as its primary instructional tool, it was categorized as plugged. If an intervention did not use technology as its primary mode of instruction, it was categorized as unplugged.

Step 4: Study Design. The design of each study was analyzed through *a priori* coding using the three types of educational research design described by the NRC (2002): Descriptive, Causal, and Mechanism. Descriptive studies describe population characteristics, simple relationships, and localized educational settings, while causal studies provide evidence of causal relationships (NRC, 2002). Mechanism studies endeavor to understand the mechanisms by which one variable exerts influence on the other (NRC, 2002).

Step 5: Study Context. Regarding the context, we attended to the locus (e.g., In-School, Out of School, and Teacher Education), participant age, and the nature of the intervention. We employed *a priori* codes drawn from the CT-MS taxonomy (Weintrop et al., 2016) to examine the target CT practices of an intervention (see Table 1). In the analysis, we differentiated studies that focused on a single category of CT-MS practice from studies that employed CT-MS practices from multiple categories. If the intervention included only practices found in the Computational Problem Solving (CPS) category of the CT-MS, the study was coded as CPS. Interventions including CT practices in both the CPS category and another CT-MS category were coded as CPS+1.

Step 6: Measures. We analyzed the types of instruments researchers utilized to collect data and report findings. All identified measures were grouped into three types: qualitative, quantitative, and mixed. We also coded the constructs that researchers sought to assess and then collapsed them into two categories: cognitive constructs (e.g., content learning, proficiency in CT skills) and affective constructs (e.g., attitudes toward CS/CT, engagement, self-efficacy). Ultimately, six categories were finalized: Quantitative cognitive, Quantitative affective, Qualitative cognitive, Qualitative affective, Mixed cognitive, and Mixed affective (see Table 2). Additionally, multiple studies provided a qualitative accounting of their work. Thus, the seventh category, Qualitative Report, was included.

Table 2

Categorization Parameters for Categories Represented in the Computational Thinking Conception Design Map

Conception	Purpose	Approach	Design	Context	Measures
Code-Centric - Description centered on programming skills and Practices. - Emphasis on skills and concepts such as algorithms, abstraction, modularization, debugging, parallelization, loops, and conditionals. - Learning objectives drawn from CSTA (2011), Barr and Stephenson (2011), or Brennan & Resnick (2012).	Test & Develop CT Tools and Programs - Testing professional development program efficacy. - Testing theoretical frameworks. - Developing instructional tools and programs. - Developing strategies for schoolwide CT integration.	Plugged - Intervention uses technology in any form. Unplugged - Intervention does not use any form of Technology.	Description - Describe populations, characteristics, simple Relationships, and localized Educational settings. - Includes descriptions of unique tools, strategies, and programs. Cause - Seeks to link one variable to another through a controlled experiment. Mechanism - Identify or describe mechanisms that link one variable to another.	In-School - Study is set in school during school hours. Out of School - Study is set outside of traditional school hours. Teacher Education - Study is set in the context of either preservice or inservice teacher professional development.	Quantitative Cognitive - Quantitative measures of cognitive change. Quantitative Affective - Quantitative measure of change in attitude, confidence, or engagement. Qualitative Cognitive - Qualitative description of cognitive change. Qualitative Affective - Qualitative description of cognitive change. Mixed Cognitive - Quantitative and qualitative measures of cognitive change. Mixed Affective - Quantitative and qualitative measures of change in attitudes, confidence, or engagement. Qualitative Report - Qualitative report of study findings.
Interdisciplinary Practice (IDP) - Focus on CT as an interdisciplinary skill That is broadly-applicable and necessary for all people. - Rarely focus on programming concepts. - Rarely draw learning objectives from a specific CT framework.	CT Assessment Development - Develop and test measures of CT learning. CS/CT Attitudes and Engagement - Change perceptions of and Attitudes towards CT and CS - Change confidence in using CT and CS. - Enhancing student engagement.				
	CT Educational Strategies - Testing pedagogical approaches to teaching CT CT-Related Cognition - Higher-order Thinking - Perceptions of CT - CT ability/student characteristic correlations				

Step 7: Synthesis of the Computational Thinking Conception Design Map (CTCDM). To facilitate the identification of the relationships between CT conceptualizations and CT operationalizations, the analysis results from steps one through six were synthesized in the CTCDM (Figure 1). The synthesis should be read from left to right, with a column for each of the six characteristics used in our analysis of the selected literature. Each paper traces a path through a node under each column. Note, while we use the term “path” in our description, we do not imply that we conducted a statistical path analysis. Rather, we created a visualization to identify trends in the corpus. For each node, the number of studies included in that category has been included as a point of reference. Please note, a small number of studies were placed in multiple categories under the same dimension. Thus, the sums of the studies under Purpose and Measures are higher than the eighty studies analyzed for this review. As an example, consider that Orton et al. (2016) sought to both teach CT skills and change students’ attitudes towards CT and CS and employed both quantitative cognitive and quantitative affective measures to estimate the intervention’s impact. Also, some studies did not contain sufficient information to be categorized as plugged/unplugged. To help further visualize the number of papers that share common characteristics and to highlight trends in the corpus, the line weights of the connections and the font sizes of the text in the nodes are proportional to the number of papers that share two connected nodes.

Further insight into trends in contexts and interventions are provided through the bars beneath the *Teach CT Skills* Purpose and *In School* and *Out of School* Context. The bar beneath the CT Skills Purpose category depicts the number of interventions that taught CT skills in conjunction with core content and the number of interventions that only taught CT skills. The bars labeled ES-MX beneath the Context categories visualize the numbers of studies in that category carried out with elementary, middle school, high school, and mixed-age students and the number of studies categorized as CPS relative to the number categorized as CPS+1.

Figure 1. Computational Thinking Conception - Research Design Map with study measures

Step 8: Validation of the Analysis Sample. A review of the titles and abstracts in the *validation sample* against the Categorization Parameters in Table 2 showed that a select number of these parameters could be reliably coded in the validation sample, namely: Plugged v. Unplugged, Student (both In-School and Out-of-School) v. Teacher Education, In-School v. Out-of-School (for Student studies). Additionally, whether students were asked to program (code) was analyzed in both samples. This allowed us to compare the *analysis sample* against the *validation sample* through frequency counts of these parameters and a Chi-squared test. The results showed that there was no significant difference between the *analysis sample* and the *validation sample* in counts for any of these parameters: Plugged v. Unplugged ($X^2(1, N = 285) = 0.038, p = 0.884$), Student (both In-School and Out-of-School) v. Teacher Education ($X^2(1, N = 315) = .001, p = 0.999$), In-School v. Out-of-School ($X^2(1, N = 228) = 3.50, p = 0.061$), and Code v. Not Code ($X^2(1, N = 276) = 2.69, p = 0.101$). In order to assess the risk of a Type II error, a categorical power analysis using G*Power (Faul, Erdfelder, Buchner, & Lang, 2009) was conducted. Using $N = 228$, $\alpha = 0.05$ and $w = 0.34$, we had adequate power (> 0.95) to confidently accept the null hypothesis. Based on this statistical analysis, we concluded that the *analysis sample* was representative of the population of English-language, peer-reviewed, high-impact empirical studies published on CT as of the end of 2019.

2.2.2. CT intervention analysis. To understand CT operationalizations, we used the CT-MS taxonomy (Weintrop et al., 2016) as an analytic framework to code instructional interventions. We worked under the assumption that practices like data use, systems thinking, and modeling are sufficiently universal to be found outside of the disciplines of math and science. Each intervention was, first, decomposed into its constituent parts. Next, each constituent part was labeled using codes drawn from the four CT-MS taxonomic categories and then classified according to a specific practice within the broader category. We note that the act of *programming* may, inherently, engage students in other CPS practices (e.g. troubleshooting and debugging or creating abstractions). For the purpose of consistent analysis, however, we constrained our coding to include only practices explicitly mentioned by authors in their manuscripts and chose not to make inferences about other practices that might be engaged through the activity of programming. Table 3 provides an example coding of Sengupta et al. (2013).

Table 3

Sample Coding of One of the Selected Studies (Sengupta et al., 2013)

CT-MS Category	CT-MS Practice	Intervention Task Activity
Modeling and Simulation	Constructing computational models	Students asked to diagrammatically represent the temporal trajectory of a ball dropped from the same height on earth and the moon.
	Using computational models To understand a concept	Students built models of ecosystems to understand relationships within the systems.
Systems Thinking	Investigating a complex system as a whole	Students constructed models to understand the cycling of matter and sustainability of ecosystems.

	Understanding the relationships within a system	Student models investigated relationships between ecosystem inhabitants.
	Thinking in levels	Students investigated ecosystems at both the micro level (bacteria) and macro level (Fish, Duckweed, Sustainability).
Data Practices	Visualizing data	Students generated graphs of speed versus time for balls dropped under differing gravity strengths (Earth vs. Moon).
Computational problem solving	Programming	Through building their models, students were introduced to agents, conditionals, loops, and variables.

3. Results

3.1. The Nature of the CT Conceptualizations Guiding Current CT Education Research

To present our results regarding the nature of CT conceptualizations guiding contemporary CT education research, we first describe salient features of CT conceptualizations emerging from our analysis, then discuss how those conceptualizations are operationalized in the studies reviewed.

Section 3.1.1. Conceptualizations of CT. Nearly all articles analyzed referenced a perceived lack of consensus in CT literature relating to the definition and scope of CT. Despite the cited lack of consensus, two categories of CT conceptualizations emerged from our analysis: *Code-centric Skills* and *Interdisciplinary Practices*. Additionally, we found that a handful of documents form the theoretical foundations of the majority of the studies analyzed. These papers include: *Mindstorms* by Seymour Papert (1980), *Computational Thinking* by Jeannette Wing (2006), the National Research Council frameworks (2010, 2012), and Grover & Pea's (2013) review.

CT as code-centric skills. Thirty-nine of eighty manuscripts reviewed (49%) were categorized as *Code-centric* because they primarily described CT in relation to programming. In their descriptions of CT and identification of essential CT knowledge, these manuscripts employed terms such as “algorithm”, “abstraction”, “modularization”, “debugging”, “parallelization”, “loops”, and “conditionals”. The same practices were also, frequently, the instructional targets of these studies. Manuscripts in this category often drew instructional and assessment objectives from CT frameworks by the Computer Science Teachers' Association (2011), Barr and Stephenson (2011), and Brennan and Resnick (2012). Because this conceptualization tends to focus on discrete programming concepts that can be learned and directly assessed, we have chosen to use the term “skills”.

CT as an interdisciplinary practice (IDP). Forty-one manuscripts (51%) primarily described CT as an interdisciplinary practice. While all eighty manuscripts referenced at least one of Wing's (2006; 2008; 2010) articles, manuscripts placed in the IDP category leaned more heavily on Wing's (2006) work for descriptions of the nature and importance of CT and to identify key CT practices. Studies in this category tended to describe CT as an “interdisciplinary practice” that is “broadly applicable” across a range of scenarios, and infrequently highlighted or assessed concepts such as loops, parallelization, conditionals, and modularization. Additionally, this group of manuscripts emphasized terms like “problem-solving”, “systems”, “creativity”,

“interdisciplinary”, and “broadly-applicable” in their descriptions of CT and arguments for the importance of CT. Because these conceptualizations emphasize CT as an approach to problem-solving that requires more than skills, we have chosen to use the term “practices”.

Section 3.1.2. Operationalizations of the CT construct. For the purpose of this analysis, we use the terms “operationalize” and “operationalization” to denote the implementation of researchers’ CT conceptualizations through the design of their studies and the content of their instructional interventions.

Common operationalizations through CT instruction. Our analysis indicates that the current body of CT education research is largely built upon code-centric CT learning activities. Through their interventions, many CT scholars have implemented CT instructional strategies more in line with traditional computer science education than with broad, interdisciplinary CT integration. Indeed, 65 of the 80 (81%) studies analyzed used programming as the primary method for teaching CT. Additionally, our study reveals three patterns in the types of CT instruction studied. First, the majority (89%) of interventions employed “plugged” educational activities. Most commonly, these interventions focused on programming, robotics, and game construction. Only six of the reviewed studies (8%) taught CT using “unplugged” approaches. Second, the majority of research interventions focused on tasks classifiable under the Computational Problem Solving (CPS) category of the CT-MS (Weintrop et al., 2016). While forty-nine percent (39/80) of the studies conceptualize CT in code-centric terms, one hundred percent operationalize CT using at least one instructional task classified as CPS. Of these CPS-focused interventions, some attempted to embed CT skills in content transmission (e.g. Boticki, Pivalica, & Seow, 2018; Lee & Soep, 2016; Peel, Fulton, & Pontelli, 2015; Sengupta et al., 2013; Sabitzer, Demarle-Meusel, & Jarnig, 2018), while many focused singularly on teaching students to write code (e.g. Atmatzidou & Demetriadis, 2016; Denner, Werner, Campe, & Ortiz, 2014; Djambong & Freiman, 2016; Leonard et al., 2016; Portelance & Bers, 2015; Pugnali, Sullivan, & Bers, 2017; Rijke, Bollen, Eysink, & Tolboom, 2018; Zhong, Wang, Chen, & Li, 2016).

Third, interventions rarely combined CPS practices with other CT-MS practices. Fifty-five (69%) studies employed intervention activities only classifiable as CPS; Twelve (15%) studies’ activities were classified as CPS+1 (e.g., Adler & Kim, 2018; Berland & Wilensky, 2015; Bower, Wood, Lai, Howe, & Lister, 2017; Chaudhary, Agrawal, Sureka, & Sureka, 2017; Chen et al., 2017; Daily, Leonard, Jörg, Babu, & Gundersen, 2017; Orton et al., 2016; Portelance & Bers, 2015; Shen, Chen, Barth-Cohen, Jiang, & Eltoukhy, 2017; Vakil, 2014), eight (10%) studies’ activities were classified as CPS + 2 (e.g., Basawapatna, 2016; Fronza, Ioini, & Corral, 2017; Jun, Han, Kim, & Lee, 2013; Repenning et al., 2015; Thomas, Odemwingie, Saunders, Watlerd, & Carlette Odemwingie, 2015; Wu, 2018), and five (6%) studies combined all four categories of CT practices (e.g., Lee & Soep, 2016; Sengupta et al., 2013; Seoane Pardo, 2018). Lastly, *Programming* was the most frequently employed among the seven CT-MS CPS practices. Sixty-five (81%) studies employed interventions that include programming. The next most frequently incorporated practice was *Troubleshooting* and *Debugging* at thirty-seven (46%) interventions, and the least frequently employed CPS practice was *Choosing effective computational tools*, which appeared in only five (6%) interventions (see Figure 2).

Figure 2. Number of occurrences of each Computational Problem Solving practice (Weintrop et al., 2016) across all studies.

Common operationalizations through research designs. As noted by Kuhn (1962), mature fields of research should exhibit substantial consistency in the types of questions pursued and research designs employed. To understand whether or not the work of the CT community exhibits this sort of coherency, we used the CTCDM to examine both the most and least common forms of CT education research. To illustrate the *most* and *least* common forms of CT education research, we describe one study that demonstrates the most common elements and another study that demonstrates the least common elements. In addition, we identified emerging patterns in the CTCDM as a whole that provide insights into the body of CT education literature analyzed for this study.

The most common form of CT education research. Boticki et al.’s (2018) article embodies the most common study elements shown in the CTCDM. We identify alignment between their paper and the following categories of the CTCDM: (1) Conception: Though they mention multiple CT conceptualizations in their introduction, a code-centric conception of CT is indicated by their statement that, “This paper aims at discussing the use of key CT concepts as proposed within Brennan and Resnick’s framework (Brennan & Resnick, 2012) with young primary school learners, with a special emphasis on the computational concepts dimension” (p. 1). Also, the authors highlight concepts such as *sequences*, *loops*, and *parallelism*. Thus, we have placed this article in the code-centric conceptualization category. (2) Purpose: Boticki et al. examined students’ development of CT concepts while CT was taught through subject area content such as mathematics or science. Given the focus of their intervention on teaching CT concepts, we have placed this study in the *Teach CT Skills* category. (3) Tools: This study was coded as *Plugged* because the intervention took place on digital tablets using a tool built on the Blockly.js framework. (4) Design: This study employed a descriptive study design (NRC, 2002) to understand students’ interactions with and learnings from the intervention. Since this study did not include any pre/post measurements, group comparisons, causal inferences, or mechanism examinations, it

was categorized as descriptive. (5) Context: This study was conducted in a school setting during school hours with elementary school age students. In addition, the study intervention focused on students practicing only CT skills within the Computational Problem Solving category of the CT-MS taxonomy (CPS); in this case, Programming. (6) Measures: Boticki et al. employed a quantitative measure of cognitive skills (Task completion times and Successful task completions) to assess the effectiveness of their intervention.

The least common form of CT education research. Because there is not a single study that demonstrates all of the least common elements, we have chosen to present a unique study that demonstrates most of the least common elements: Seoane Pardo (2018). (1) Conception: Seoane Pardo (2018) adopted an IDP conceptualization with the assertion that CT represents “the way humans resolve problems and take decisions” (p. 21) and his work focused on developing an approach for teaching students to apply CT to ethical conundrums. (2) Purpose: The purpose of this study was to demonstrate how CT practices can be used to help students “approach, explain and try to resolve” (p. 33) moral dilemmas. Because of its focus on pedagogy, we have categorized this study as *CT Education Strategies*. (3) Tools: While some electronic resources (i.e. videos and an online simulator) were used to support instruction, the primary instructional activities were conducted without technology; earning this study a rare *Unplugged* designation. (4) Design: Given the study’s demonstrational nature, we categorized the Design of this work as *Description*. (5) Context: This study was conducted with high school students during school hours and was one of five works that used CT practices from all CT-MS categories (CPS+1). (6) Measures: Seoane Pardo used qualitative measures of both cognitive and affective constructs to understand the effectiveness of his approach.

Section 3.1.3. Study design insights from the synthesis. Our analysis of the CTCDM in its entirety reveals several patterns in the body of CT education research analyzed for this study. First, the CT community’s continual refrain about the lack of a consensus definition of CT is supported by the near even split between code-centric and IDP conceptualizations of CT. In spite of this conceptual division, CT’s historical roots in computer science are clearly displayed through the pervasive use of plugged, programming-based approaches to teaching CT. Even bearing these roots in mind, the absence of unplugged instruction is striking. As discussed by Kuhn (1962), paradigm (i.e. Conceptualization) influences research design (i.e. Operationalization). Hence, one would expect that researchers working in a discipline that is rooted in computer science would design CT interventions that present CT as its own, unique content through the core instructional tool of computer science: coding. It similarly follows that these studies would assess the efficacy of their intervention with measurement instruments based on students’ ability to write and understand coding concepts. This CS-roots/CT-instruction connection is underscored by our finding that 81% of the interventions analyzed taught students to write code, and only 50% of the interventions attempted to teach CT skills in conjunction with academic content.

Second, only a single study (Berland & Wilensky, 2015) pursued the mechanisms through which students come to understand CT practices. Each of the other seventy-nine studies were either descriptive or causal (NRC, 2002), with a heavy inclination towards descriptive research designs (61% of all studies). While we do not offer speculation on the cause of the imbalance in the distribution of studies across the three research types, we acknowledge that this significant gap in the literature offers opportunities for researchers interested in pursuing both causal relationships and the underlying mechanisms that promote student CT development. Third, we note the prevalence of quantitative cognitive measures of intervention effectiveness (i.e., “what happened”). While these quantitative measures provide insight into the cognitive results of an

intervention, they yield scarce evidence of the operations of the mechanisms responsible for any gains (i.e., “why did it happen”). Finally, the paucity of literature focusing on efforts to support teachers in including CT in their classrooms is notable. Only 19% of our sample reports programs for developing preservice and inservice teachers’ understanding of CT or confidence in bringing CT to their classroom. As noted by Margolis Estrella, Goode, Home, and Nao (2010), Yadav and colleagues (2016, 2017), and others, teachers will not be able to provide their students with quality CT instruction until the teachers themselves understand the construct and feel confident about its integration in their curriculum.

Section 3.2. Strategies for Integrating CT in Core Curriculum

Only twelve (15%) of the studies reviewed explicitly investigated integrating CT with subject-area content. We highlight that all instances of CT/content integration (CTCI) that included measures of student learning reported positive outcomes and we report our findings related to strategies for CTCI according to the four categories of CT practices described by Weintrop and colleagues’ (2016) CT-MS framework: beginning with Data Practices. Instances of Data Practice tasks were found across multiple subjects. One example of data collection in the context of art instruction is Rode and colleagues’ (2015) description of Computational Making. In this study students engaged in CT by using e-textiles to make plush monsters laced with conductive thread and instrumented with sensors programmed using Arduino. Describing the activity, Rode et al. identify multiple instances of students gathering data from manifestations of their code (blinking lights or sounds) and revising their programs to achieve desired outcomes.

Regarding math and science, opportunities to engage with data were myriad. Orton and colleagues (2016) demonstrate the benefits of both CTCI and repeated CT exposure, and mention a series of lessons that integrate CT with mathematics and science content through topics like “US census data, radioactivity, black holes, and video games” (p.707). In addition, Weintrop et al. (2016) recount a lesson where students use the video game Angry Birds and Physics Tracker (<https://physlets.org/tracker/>) to learn principles of motion. Yadav and colleagues (2016) suggest that environmental science classes might use Google Public Data Explorer (<http://www.google.com/publicdata>) to collect and compare data on countries’ greenhouse gas emissions.

The second category of CT-MS practices is Modeling and Simulation. Most of the examples of integrated Modeling and Simulation were found in the context of using Agent-Based Modeling (ABM) to simulate scientific phenomena (Basawapatna, 2016; Repenning et al., 2015; Sengupta et al., 2013). The most notable example in this category is Repenning and colleagues’ (2015) description of the Scalable Game Design (SGD) program. SGD teaches students basic programming and modeling concepts through designing games using AgentSheets and AgentCubes (Agentsheets, 2018). Students then apply these skills to the construction of scientific simulations. The SGD approach to CT/science integration has been shown to be scalable, to enhance student motivation, to broaden CS exposure and participation, and to lower barriers to CT integration. Additionally, Sengupta et al. (2013) and Weintrop et al. (2016) describe lessons where students construct simulations to understand interactions within systems. Sengupta and colleagues (2013) explain the use of CTSiM for teaching physics through modeling kinematics and environmental science through modeling interactions in an ecosystem, and provide evidence of enhanced CT and content learning. Weintrop et al. (2016) describe students using PhET (University of Colorado, 2018) to explore atomic interactions. Basawapatna’s (2016) work points to the potential of pattern-based programming approaches to ease integration of CT practices into

science content. Outside of the sciences, Yadav et al. (2016) proposes CT/social studies integration through lessons that requires students to explore the relationships between individual housing preferences and the development of racial segregation over time, and Sabitzer et al. (2018) explain the use of modeling and diagrams in foreign language lessons.

As previously noted, all of the studies reviewed taught CT practices classified under the category of Computational Problem Solving. Regarding specific CT integration strategies, Jenkins' (2015) description of students using Scratch to build poem-generating programs provides unique insight into a possible integration of CT into English language arts and shows that students demonstrated both enhanced content and CT understanding. In Basawapatna's study (2016) students used the Simulation Creation Toolkit to construct predator/prey simulations. Vakil (2014) uses analysis of systems that create socioeconomic inequalities to engage minority youth in computational thinking while learning social studies content. While Peel, Sadler, and Friedrichsen (2019) report the efficacy of their CTUPDATE framework for teaching natural selection through students creating unplugged algorithms, Niemelä, Partanen, Harsu, Leppänen, and Ihantola (2017) use data drawn from teacher artifacts to propose a CT learning trajectory in mathematics classrooms. Manuscripts within this category also included exemplars of CTCI through mathematics in elementary school (Falloon, 2016) and one of the few examples of CT/Art integration (Rode et al., 2015). Notably, Falloon's (2016) work provides evidence of integrated CT eliciting higher-order thinking.

Descriptions for or tests of lessons containing Systems Thinking practices were found in manuscripts by Basawapatna (2016), Henrique de Paula, Burn, Noss, and Valente (2018), Lee and Soep (2016), Rode et al. (2015), Sabitzer et al. (2018), Sengupta et al. (2013), Seoane Pardo (2018), Vakil (2014), Weintrop et al. (2016), and Yadav et al. (2016). The only study that included all of the Systems Thinking practices was Vakil (2014), and many of the other examples of systems thinking were found in the context of artifact construction; whether it be a simulation (Basawapatna, 2016; Sengupta et al., 2013), game (Henrique de Paula et al., 2018), or plush toy (Rode et al., 2015). Interestingly, working with systems of social and economic inequality or moral dilemmas seem to provide especially rich opportunities for CTCI (e.g. Lee and Soep, 2016; Seoane Pardo, 2018; Vakil, 2014). Finally, activities that include systems thinking seem to be more complex than activities that do not include systems thinking practices. Most of the studies with curricular examples in the systems thinking category also frequently included practices identified in the other three categories of CT-MS practices (CPS+1).

4. Discussion

Our analysis of the selected studies indicates that the CT conceptualizations of scholars are divided into two categories: CT as a *code-centric skill* and CT as an *interdisciplinary practice (IDP)*. Whether adopting a code-centric or IDP conceptualization, the CT research community primarily operationalizes the CT construct in the form of plugged, programming-based activities, with relatively few attempts to integrate CT into content-area curriculum. In addition, our synthesis of the CTCDM reveals that the most common form of CT educational intervention adopts an IDP conceptualization, attempts to teach CT skills (often without curricular content), utilizes a descriptive design, occurs in the context of a school, and employs quantitative cognitive measures of the intervention's efficacy. Several of the prominent study design characteristics (i.e., descriptive designs, in-school contexts, and quantitative cognitive measures) are sensible given our selection criteria of empirical studies conducted with K-12 populations and the "young" nature

of CT education literature. There were, however, results of our analysis that warrant further exploration. Specifically, we highlight three issues in our discussion: (1) tension between CT conceptualizations and CT operationalization, (2) a dearth of research relating to unplugged approaches to teaching CT, and (3) the need for research on CT professional development for teachers.

First, while theoretically and conceptually building their work using common authors and frameworks (e.g., Brennan & Resnick, Barr & Stephenson, CSTA, Wing), the studies reviewed continually state that the CT education community has not arrived at a consensus definition of CT. Our conceptualization analysis clearly shows that the reiterated lack of consensus regarding the nature of CT does, indeed, exist. Despite divided conceptualizations of CT, however, there appears to be consensus on the operationalization of CT through programming. Although we found a 49/51 conceptualization split between CT as a code-centric skill and CT as an interdisciplinary practice, all of the interventions operationalized CT using tasks categorized under Weintrop et al.'s (2016) Computational Problem Solving category and 81% of the studies used programming as the primary means of instruction. Thus, we point to the potential for a problematic mismatch between CT conceptualizations and CT operationalizations; that, as noted by Hestness et al. (2018), could complicate efforts to integrate CT with disciplinary content.

Our findings are in line with other authors who have noted the code-centric focus of most CT instruction and the misperception that CT equals programming (Lu & Fletcher, 2009; Shute et al., 2017; Wing, 2017). Indeed, Haseski et al.'s (2018) recent analysis of 59 CT definitions highlights the prominence of technology and programming in CT definitions, despite frequent assertions that CT is for everyone “not only for those who are computer science professionals” (p. 36). Reinforcing this finding, Özcınar (2017) asserts that the bulk of CT studies published to date have appeared in journals and conference proceedings of CS education. Given the above, it is clear that CT's CS roots are the guiding force behind CT education and research initiatives. The present lack of a consensus CT framework which could chart a path towards CT operationalization beyond programming results in the absence of a counterbalance against the pull of CT's computer science roots. Consequently, interdisciplinary conceptualizations that take an operational approach outside programming situate themselves in the “innovative fringe,” rather than the mainstream of the literature.

Second, the lack of studies addressing unplugged approaches to CT education is striking, but expected. As noted by Moreno-León (2018), precious little work has been conducted to understand the affordances of CT unplugged. Nevertheless, our review points to exciting examples of unplugged activities as effective tools to promote student CT and to integrate CT with subject-area content (e.g., Seoane Pardo, 2018; Peel et al., 2019). Additionally, the studies included in our analysis show that unplugged CT activities can both ease integration of CT with disciplinary content and provide learning experiences that are especially computationally rich (i.e. they employ CT-MS practices beyond the Computational Problem Solving category).

Lastly, our review revealed a significant need for research on CT professional development programs (CT-PD) for preservice and in-service teachers. The relatively low number of CT-PD studies captured in our sample is consistent with the findings of Voogt et al. (2018) and is concerning given that teachers often have limited understanding of CT as a practice in core subject areas (Authors, in review; Ling, Saibin, Labadin, & Aziz, 2018). If students are to become computationally literate, they must have adequately prepared teachers (Cuny, 2011; Yadav et al., 2017). As our findings indicate, however, the bulk of CT research activity to date has focused on students learning and practicing CT.

With respect to our second research question centering on CT integration, several salient patterns emerged. The studies analyzed yield compelling evidence for the proposed synergies between CT and core content, whereby teaching CT practices and content together enhances the apprehension of both (Rich, Binkowski, Strickland, & Franklin, 2018; Sengupta et al., 2013). While the largest proportion of the studies analyzed were situated in traditional school settings, very few focused on teaching CT in the context of core academic curricula. This suggests the need for more research on CTCI in the context of traditional school settings to provide evidence-based implications for best practices that are applicable to actual classrooms.

Regarding specific strategies to support CTCI, our findings suggest the following. First, involving students in work based on complex questions and systems may provide the most fruitful opportunities for developing both students' content understanding and CT proficiency. Most of the studies containing activities that included at least one skill from each of the categories of the CT-MS (CPS+1) involved building simulations of ecosystems (Sengupta et al., 2013), addressing systems that lead to inequality (Lee & Soep, 2016; Vakil, 2014), or confronting moral dilemmas (Seoane Pardo, 2018). We highlight exemplars from Vakil (2014) and Lee and Soep (2016) which leverage Critical Computational Literacy (CCL) (Lee & Soep, 2016) as revealing a promising strategy for engaging underrepresented groups in CT. Second, while Modeling and Simulation practices may offer the richest opportunities to develop a variety of CT skills, this category of practice was the least frequently identified in the studies analyzed. This may be the result of either lack of access to the specialized software required for this practice or, as suggested by Tatar et al. (2017), a curriculum that is not sufficiently rich to support students in thinking deeply about complex systems.

Third, while a handful of studies proposed activities for integrating CT and core content without asking students to write code, only three (Dasgupta et al., 2017; Peel et al., 2019; Sabitzer et al., 2018) actually examined the effectiveness of unplugged approaches to CT instruction. This finding points to the large literature gap surrounding unplugged approaches to CT identified by Moreno-León et al. (2018) and reveals a significant research opportunity for those interested in pursuing alternative forms of CT instruction. Lastly, of the twenty-two practices described in the CT-MS, the least frequently identified were Manipulating Data (four studies) and Defining Systems (no instances). The absence of these practices may be an indication that they are more difficult to incorporate than other CT-MS practices and that special effort may be required to begin introducing these skills to students.

Drawing on a synthesis of the most promising elements of the above-presented CTCI strategies, we make an initial proposal for some foundational elements of a *Computational Pedagogy (CP)* that supports the integration of CT with core content. First, as highlighted by Papert's (1980) seminal work focusing on the use of Logo programming to teach mathematics, CP is fundamentally constructivist in nature. Specifically, students learn both content and CT through the construction of products. Exemplars of this approach include SGD from Repenning et al. (2015), Poem Generator from Jenkins (2015), and unplugged natural selection algorithms from Peel et al. (2019). Second, as identified in our integration analysis, lessons that engage students in thinking about complex systems are integral to CP in that they provide fertile ground for students to both engage in interdisciplinary analysis of complex problems and to begin to propose solutions for these problems. Third, as discussed by scholars such as Weintrop et al. (2016), CP must provide students with opportunities to deeply engage with data through data collection, manipulation, analysis, and visualization. ABM seems to provide some of the most promising tools and strategies for accomplishing this goal. Finally, CP depends on presenting students with authentic problems

that are meaningful to the students. As discussed by Lee and Soep (2016) and Vakil (2014), this approach dramatically enhances student interest, engagement, and buy-in. To recap, we propose that *Computational Pedagogy* is constructivist in orientation, engages students in systems thinking through complex problems, includes extensive work with data, and is built on contexts and problems that are personally meaningful to students.

5. Implications

The code-centric CT operationalizations that have guided CT education and research may risk depriving students of rich opportunities to fully develop foundational CT practices for two reasons. First, we argue that code-centric approaches may fail to leverage the benefits of multiple forms of representation as explained by Greeno and Hall (1997). As noted by Basu (2016), learning through multiple external representations “produces deeper understanding of science phenomena, something that is harder to achieve with single representations” (p. 67-68). Basu also mentions that this deeper understanding may help students to become more adept at forming abstractions and transferring their learning from one context to the next. New STEM frameworks (e.g., NGSS) have specifically incorporated practices such as CT as a synergistic approach to both broadening students’ exposure to science and engineering practices and to providing new epistemic tools for learning core STEM concepts. Narrowly constraining the representational forms used in CT to programming limits success on both of these fronts.

Second, a sole focus on coding may place low SES and underrepresented groups at a particular disadvantage because significant structural and social barriers to technology integration already exist in the under-resourced schools where a majority of students are from low SES and underrepresented populations (Google & Gallup, 2016). Additionally, these schools are often under pressure to improve student performance on high stakes English, math, and science exams (Repenning et al., 2015; Yadav et al., 2016). Consequently, schools are often unable to make space in the curriculum for code-centric instruction that is perceived as not being directly aligned with standardized tests and are unlikely to have the resources to offer elective courses that teach CT skills (Repenning et al., 2015; Yadav et al., 2016; Montoya, 2017).

Given these two concerns associated with the code-centric nature of CT instruction, we propose that the CT research and education communities begin to consider an expansion of the current CT paradigm (Kuhn, 1962) to allow for greater emphasis on unplugged instructional strategies that may provide additional instructional affordances. Instead of beginning with code, we believe that it may be beneficial for the CT community to consider the potential affordances of first teaching students to deeply understand the problem at hand (via systems thinking), to be aware of the data they need and can produce (via data practices), and to identify the most effective computational tools. From this foundational knowledge, students will be better situated to meaningfully engage in the process of employing computational tools (code-based or otherwise) to produce solutions to complex problems. This approach builds on the work of Lu & Fletcher (2009), who argue that coding may be an excellent capstone CT activity for those who already have CT fundamentals well in hand and has been demonstrated as effective by Hermans and Avialoglou (2017).

A potential strategy for building the CT education community’s understanding of a variety of strategies that may support meaningful CTCI could be broad adoption of a CPS+1 approach to CT education and research. The CPS+1 approach purposefully designs interventions that integrate coding with a practice or practices found in another CT-MS category. We recognize the importance

of the practices listed under the CPS category, but suggest that the CT education community evenhandedly consider whether or not other categories of CT-MS practices may be a better fit for particular curricula, contexts, teachers, or students. In essence, we contend that CT research and instruction can maximize the potential of CT by utilizing a wide array of CT practices delivered through a multitude of approaches to instruction. Thus, we underscore that there is much work to be conducted to identify and understand the benefits of non-coding approaches to CT education that may ease CTCI and lay the groundwork for more meaningful learning through code-based pedagogies.

If students are to benefit from the generative nature of CT/subject-content integration envisioned by Grover & Pea (2018), the CT/science synergies described by Sengupta et al. (2013), or the expanded access noted by Repenning et al. (2015), there will first need to be properly resourced teachers; for teachers are the intersection between the academic community and the students that researchers hope to impact. In much the same way that Constructivist pedagogy and Project Based Learning have been woven into preservice teacher preparation programs, CT instruction must become an integral component of teacher preparation. To this point, Yadav et al. (2017) provide a practical method for achieving this goal by introducing CT to preservice teachers through required instructional technology and teaching methods courses. According to Yadav et al. (2017), through partnerships between education faculty and computer science faculty, programs can be developed whereby preservice teachers learn about CT through instructional technology courses and learn to integrate CT through their methods courses. In this regard, we highlight that these initiatives must not focus solely on teaching educators to program.

6. Conclusion

In closing, we recognize the historic coupling of CT and CS, and applaud the numerous initiatives that have sought to bring CS to all students. We caution, however, that the form of CT research and educational interventions has been narrowly conceived. The code-centric, one-size-fits-all nature of the present body of CT education research runs the risk of erecting barriers to providing all students with CT education through CT integration in core academic curriculum. In recognition of the current trajectory of CT education research, we make the following recommendations for future research: (1) Investigation is needed to identify effective strategies for teaching foundational CT practices without relying on teaching students to program; (2) A research-supported set of best practices for teaching integrated CT in conjunction with core content is needed as a foundation for the development of educational programs to support teachers; (3) Building upon best practices for CT integration, substantial research is needed to provide implications for the design of both preservice and inservice teacher education programs to support CT integration. In particular, the academic community must understand the barriers that teachers encounter that hinder CT integration and strategies that help teachers overcome these barriers. In addition, a body of literature relating to effective practices for CT education for teachers will be fundamental to the development of large-scale initiatives that motivate teachers towards and support them in this work. Ultimately, preparing teachers of core content to authentically integrate CT into their curriculum is the surest path towards realization of Wing's (2006) vision of computational thinking for all.

Funding

This research did not receive grants from funding agencies in the public, commercial or not-for-profit sectors.

References

- Adler, R. F., & Kim, H. (2018). Enhancing future K-8 teachers' computational thinking skills through modeling and simulations. *Education and Information Technologies*, 23(4), 1501-1514. doi:10.1007/s10639-017-9675-1
- Agentsheets, Inc. (August 10, 2018). Agentsheets. Retrieved from <http://www.agentsheets.com/>
- Atmatzidou, S., & Demetriadis, S. (2016). Advancing students' computational thinking skills through educational robotics: A study on age and gender relevant differences. *Robotics and Autonomous Systems*, 75, 661–670. doi:10.1016/j.robot.2015.10.008
- Authors. (In Review). Secondary science teachers' understandings of the epistemic nature of science practices. *Science Education*.
- 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, 48–54. doi:10.1145/1929887.1929905
- Basawapatna, A. (2016). Alexander meets Michotte: A simulation tool based on pattern programming and phenomenology. *Journal of Educational Technology & Society*, 19(1), 277.
- Basu, S. (2016). *Fostering Synergistic Learning of Computational Thinking and Middle School Science in Computer-based Intelligent Learning Environments*. (Unpublished Doctoral Dissertation). Vanderbilt University. Nashville, TN.
- Berland, M., & Wilensky, U. (2015). Comparing Virtual and Physical Robotics Environments for Supporting Complex Systems and Computational Thinking. *Journal of Science Education and Technology*. doi:10.1007/s10956-015-9552-x
- Bers, M. U. (2018). Coding and Computational Thinking in Early Childhood: The Impact of ScratchJr in Europe. *European Journal of STEM Education*, 3(3), 08. doi: 10.20897/ejsteme/3868
- Bettis, P. J., & Gregson, J. A. (2001). The why of research: Paradigmatic and pragmatic Considerations. In E.I. Farmer & J.W. Rojewski (Eds.) *Research pathways: Writing professional papers, theses, and dissertations in workforce education* (pp. 1-21). New York, NY: University Press of America.
- Bower, M., Wood, L. N., Lai, J. W. M., Howe, C., & Lister, R. (2017). Improving the Computational Thinking Pedagogical Capabilities of School Teachers. *Australian Journal of Teacher Education*, 42(423). doi:10.14221/ajte.2017v42n3.4
- Boticki, I., Pivalica, D., & Seow, P. (2018). The Use of Computational Thinking Concepts in Early Primary School. *Proceedings of the International Conference of Computational Thinking Education 2018*.
- Brennan K, Resnick M (2012). New frameworks for studying and assessing the development of computational thinking. Presented at the American Education Researcher Association, Vancouver, Canada.
- Chaudhary, V., Agrawal, V., Sureka, P., & Sureka, A. (2016, December). An experience report on teaching programming and computational thinking to elementary level children using lego robotics education kit. In *Technology for Education (T4E), 2016 IEEE Eighth International Conference on* (pp. 38-41). IEEE. doi:10.1109/t4e.2016.016
- Chen, G., Shen, J., Barth-Cohen, L., Jiang, S., Huang, X., & Eltoukhy, M. (2017). Assessing elementary students' computational thinking in everyday reasoning and robotics programming. *Computers & Education*, 109, 162–175. doi:10.1016/j.compedu.2017.03.001

- Corbin, J., & Strauss, A. (2015). *Basics of qualitative research: Techniques and procedures for developing grounded theory* (4th ed.). Thousand Oaks, CA: Sage.
- Computer Science Teachers' Association (2011). Operational Definition of Computational Thinking for K–12 Education, 2011. Accessed from <http://www.csta.acm.org/Curriculum/sub/CompThinking.html>
- Creswell, J.W. & Poth, C.N. (2018). *Qualitative inquiry & research design: Choosing among the Five approaches* (4th ed.). Thousand Oaks, CA: Sage.
- Cuny, J. (2011). Transforming Computer Science Education in High Schools. *Computer*, 44(6), 107-109. doi:10.1109/mc.2011.191
- Daily, S. B., Leonard, A. E., Jörg, S., Babu, S., & Gundersen, K. (2014). Dancing Alice: Exploring embodied pedagogical strategies for learning computational thinking. *SIGCSE 2014 - Proceedings of the 45th ACM Technical Symposium on Computer Science Education*, 91–96. <https://doi.org/10.1145/2538862.2538917>
- Dasgupta, A., Rynearson, A. M., Purzer, S., Ehsan, H., & Cardella, M. E. (2017, June). Computational thinking in K-2 classrooms: evidence from student artifacts (fundamental). In *2017 ASEE Annual Conference & Exposition*.
- Djambong, T., & Freiman, V. (2016). Task-Based Assessment of Students' Computational Thinking Skills Developed through Visual Programming or Tangible Coding Environments. *International Association for Development of the Information Society*.
- Denner, J., Werner, L., Campe, S., & Ortiz, E. (2014). Pair Programming: Under What Conditions Is It Advantageous for Middle School Students? *Journal of Research on Technology in EducationOnline) Journal Journal of Research on Technology in Education*, 463, 1539–1523. doi:10.1080/15391523.2014.888272
- Falloon, G. (2016). An analysis of young students' thinking when completing basic coding tasks using Scratch Jnr. On the iPad. *Journal of Computer Assisted Learning*. doi:10.1111/jcal.12155
- Faul, F., Erdfelder, E., Buchner, A., & Lang, A.-G. (2009). Statistical power analyses using G*Power 3.1: Tests for correlation and regression analyses. *Behavior Research Methods*, 41, 1149-1160.
- Flórez, F. B., 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. doi:10.3102/0034654317710096
- Fronza, I., Ioini, N. El, & Corral, L. (2017). Leveraging robot programming to foster computational thinking. *CSEDU 2017 - Proceedings of the 9th International Conference on Computer Supported Education*, 2(Csedu), 109–116. <https://doi.org/10.5220/0006310101090116>
- Google, & Gallup. (2016). *Diversity Gaps in Computer Science: Exploring the Underrepresentation of Girls, Blacks and Hispanics*: Google for Education.
- Gough, D., & Thomas, J. (2012). Commonality and diversity in reviews. In D. Gough, S. Oliver, & J. Thomas (Eds.), *An introduction to systematic reviews* (2nd ed., pp. 43-70). London, Sage.
- Gough, D., Oliver, S., & Thomas, J. (Eds.). (2012). *An introduction to systematic reviews*. Sage.
- Grant, M. J., & Booth, A. (2009). A typology of reviews: an analysis of 14 review types and associated methodologies. *Health Information & Libraries Journal*, 26(2), 91-108. doi:0.1111/j.1471-1842.2009.00848.x

- Greeno, J. G., & Hall, R. P. (1997). Practicing representation: Learning with and about representational forms. *Phi Delta Kappan*, 78, 361-367.
- Grover, S., & Pea, R. (2013). Computational thinking in K-12: A review of the state of the field. *Educational Researcher*, 42(1), 38-43. doi:10.3102/0013189X12463051
- Grover, S., Pea, R., & Cooper, S. (2015). Designing for deeper learning in a blended computer science course for middle school students. *Computer Science Education*, 25(2), 199-237. doi:10.1080/08993408.2015.1033142
- Grover, S., & Pea, R. (2018). Computational Thinking: A competency whose time has come. *Computer Science Education: Perspectives on Teaching and Learning in School*, 19.
- Halevi, G., Moed, H., & Bar-Ilan, J. (2017). Suitability of Google Scholar as a source of scientific information and as a source of data for scientific evaluation—Review of the literature. *Journal of Informetrics*, 11(3), 823-834.
- Harzing, A. W., & Van Der Wal, R. (2009). A Google Scholar h-index for journals: An alternative metric to measure journal impact in economics and business. *Journal of the American Society for Information Science and Technology*, 60(1), 41-46.
- Haseski, H. İ., Ula, İ., & Tu, U. (2018). Defining a New 21st Century Skill-Computational Thinking : Concepts and Trends, 11(4), 29-42. doi:10.5539/ies.v11n4p29
- Heintz, F., Mannila, L., & Färnqvist, T. (2016, October). A review of models for introducing computational thinking, computer science and computing in K-12 education. In *Frontiers in education conference (FIE), 2016 IEEE* (pp. 1-9). IEEE.
- Henrique de Paula, B. H., Burn, A., Noss, R., & Valente, J. A. (2018). Playing Beowulf: Bridging computational thinking, arts and literature through game-making. *International Journal of Child-Computer Interaction*, 16, 39-46. <https://doi.org/10.1016/j.ijcci.2017.11.003>
- Hermans, F., & Aivaloglou, E. (2017). To Scratch or not to Scratch? *Proceedings of the 12th Workshop on Primary and Secondary Computing Education - WiPSCE '17*, 49-56. <https://doi.org/10.1145/3137065.3137072>
- Hestness, E., Ketelhut, D. J., McGinnis, J. R., & Plane, J. (2018). Professional Knowledge Building within an Elementary Teacher Professional Development Experience on Computational Thinking in Science Education. *Journal of Technology and Teacher Education*, 26(3), 411-435.
- Hirsch, J. E. (2005). An index to quantify an individual's scientific research output. *Proceedings of the National Academy of Sciences of the United States of America*, 102(46), 16569-16572. doi: 10.1073/pnas.0507655102
- Jenkins, C. (2015). Poem Generator: A comparative quantitative evaluation of a microworlds-based learning approach for teaching English. *International Journal of Education and Development Using Information and Communication Technology*, 11(2), 153-167.
- 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). In *future directions in computer science education summit meeting, Orlando, FL*.
- Jun, S. J., Han, S. G., Kim, H. C., & Lee, W. G. (2014). Assessing the computational literacy of elementary students on a national level in Korea. *Educational Assessment, Evaluation and Accountability*. doi:10.1007/s11092-013-9185-7
- Kuhn, T. S. (1962). *The Structure of Scientific Revolutions*. University of Chicago Press. Chicago, IL.

- Lee, C., & Soep, E. (2016). None but Ourselves Can Free Our Minds: Critical Computational Literacy as a Pedagogy of Resistance. *Equity & Excellence in Education*. doi:10.1080/10665684.2016.1227157
- Leonard, J., Alan Buss, B., Ruben Gamboa, B., Monica Mitchell, B., Olatokunbo Fashola, B. S., Tarcia Hubert, B., & Sultan Almuhyirah, B. (n.d.). Using Robotics and Game Design to Enhance Children's Self- Efficacy, STEM Attitudes, and Computational Thinking Skills. doi:10.1007/s10956-016-9628-2
- Li, W.-L., Hu, C.-F., & Wu, C.-C. (2016). *Teaching High School Students Computational Thinking with Hands-on Activities*. Paper presented at the Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education, Arequipa, Peru. doi: 10.1145/2899415.2925496
- Ling, U. L., Saibin, T. C., Labadin, J., & Aziz, N. A. (n.d.). Preliminary Investigation: Teachers' Perception on Computational Thinking Concepts. *Journal of Telecommunication, Electronic and Computer Engineering*, 9 (2), 2289–8131.
- Lu, J. J., & Fletcher, G. H. (2009). Thinking about computational thinking. *ACM SIGCSE Bulletin*, 41(1), 260-264. doi:10.1145/1539024.1508959
- Margolis, J., Estrella, R., Goode, J., Holme, J. J., & Nao, K. (2010). *Stuck in the shallow end: Education, race, and computing*. MIT Press. Cambridge, MA.
- Martín-Martín, A., Orduna-Malea, E., Thelwall, M., & López-Cózar, E. D. (2018). Google Scholar, Web of Science, and Scopus: A systematic comparison of citations in 252 subject categories. *Journal of Informetrics*, 12(4), 1160-1177.
- Montoya, A. (n.d.). Computer Science for All : Opportunities Through a Diverse Teaching Workforce. *Harvard Kennedy School of Hispanic Policy*. (29), 47-62. The Harvard Kennedy School.
- Moreno-León, J., Román-González, M., & Robles, G. (2018). On computational thinking as a universal skill: A review of the latest research on this ability. In *Global Engineering Education Conference (EDUCON), 2018 IEEE* (pp. 1684-1689). IEEE. doi:10.1109/educon.2018.8363437
- National Research Council. (2002). *Scientific research in education*. National Academies Press.
- National Research Council (2010) Report of a workshop on the scope and nature of computational thinking. The National Academies Press, Washington, DC
- National Research Council (2012a) A framework for K-12 science education: practices, crosscutting concepts, and core ideas. National Academies Press, Washington, DC
- NGSS Lead States. 2013. *Next Generation Science Standards: For States, By States*. The National Academies Press. Washington, DC.
- Niemel, P., Partanen, T., Harsu, M., Leppänen, L., & Ihantola, P. (2017). Computational thinking as an emergent learning trajectory of mathematics. *ACM International Conference Proceeding Series*, (November), 70–79. <https://doi.org/10.1145/3141880.3141885>
- Orton, K., Weintrop, D., Beheshti, E., Horn, M., Jona, K., & Wilensky, U. (2016). Bringing Computational thinking into high school mathematics and science classrooms. *ICLS 2016 Proceedings*. Singapore: International Society of the Learning Sciences.
- Özçınar, H. (2017). Bibliometric analysis of computational thinking. *Educational Technology Theory & Practice*, 7(2), 149-171.
- Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. Basic Books, Inc. New York, NY.
- Peel, A., Fulton, J., & Pontelli, E. (2015). DISSECT: An experiment in infusing computational

- thinking in a sixth grade classroom. *Proceedings - Frontiers in Education Conference, FIE, 2014*. doi:10.1109/FIE.2015.7344240
- Peel, A., Sadler, T. D., & Friedrichsen, P. (2019). Learning natural selection through computational thinking: Unplugged design of algorithmic explanations. *Journal of Research in Science Teaching*, 56(7), 983-1007.
- P. Pitol, S., & L. De Groote, S. (2014). Google Scholar versions: do more versions of an article mean greater impact?. *Library Hi Tech*, 32(4), 594-611.
- Portelance, D. J., & Bers, M. U. (n.d.). Code and Tell: Assessing Young Children's Learning of Computational Thinking Using Peer Video Interviews with ScratchJr. doi:10.1145/2771839.2771894
- Pugnali, A., Sullivan, A., & Bers, M. U. (2017). THE impact of user interface on young children's computational thinking. *Journal of Information Technology Education: Innovations in Practice*, 16(1), 171-193. <https://doi.org/10.28945/3768>
- Repenning, A., Webb, D. C., Han Koh, K., Nickerson, H., Miller, S. B., Brand, C., Horses, I., Basawapatna, A., Gluck, F., Grover, R., Gutierrez, K., Repenning, N. (2015). Scalable Game Design: A Strategy to Bring Systemic Computer Science Education to Schools through Game Design and Simulation Creation. *ACM Transactions on Computing Education*, 15(11). doi:10.1145/2700517
- Rich, K. M., Binkowski, T. A., Strickland, C., & Franklin, D. (2018, August). Decomposition: A K-8 Computational Thinking Learning Trajectory. In *Proceedings of the 2018 ACM Conference on International Computing Education Research*(pp. 124-132). ACM. doi: 10.1145/3230977.3230979
- Rijke, W. J., Bollen, L., Eysink, T. H. S., & Tolboom, J. L. J. (2018). Computational Thinking in Primary School: An Examination of Abstraction and Decomposition in Different Age Groups. *Informatics in Education*, 17(1), 77-92. doi:10.15388/infedu.2018.05
- Rode, J. A., Booker, J., Marshall, A., Weibert, A., Aal, K., von Rekowski, T., ... Schleeter, A. (2015). From computational thinking to computational making. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2015 ACM International Symposium on Wearable Computers - UbiComp '15*. doi:10.1145/2800835.2800926
- Rodriguez, B., Kennicutt, S., Rader, C., & Camp, T. (2017). Assessing computational thinking in CS unplugged activities. *Proceedings of the Conference on Integrating Technology into Computer Science Education, ITiCSE*, 501-506. <https://doi.org/10.1145/3017680.3017779>
- Royal Society. (2012). Shut down or restart: The way forward for computing in UK schools. Retrieved from <http://royalsociety.org/education/policy/computing-in-schools/report/>
- Sabitzer, B., Demarle-Meusel, H., & Jarnig, M. (2018). Computational thinking through modeling in language lessons. *IEEE Global Engineering Education Conference, EDUCON, 2018-April(2)*, 1913-1919. <https://doi.org/10.1109/EDUCON.2018.8363469>
- Sengupta, P., Kinnebrew, J. S., Basu, S., Biswas, G., & Clark, D. (2013). Integrating computational thinking with K-12 science education using agent-based computation: A theoretical framework. *Education and Information Technologies*. doi:10.1007/s10639-012-9240-x
- Shen, J., Chen, G., Barth-Cohen, L., Jiang, S., & Eltoukhy, M. (2016). Developing a Language-neutral Instrument to Assess Fifth Graders' Computational Thinking. *Transforming Learning, Empowering Learners*.

- Shute, V. J., Sun, C., & Asbell-Clarke, J. (2017). Demystifying computational thinking. *Educational Research Review*, 22, 142–158. doi:10.1016/j.edurev.2017.09.003
- Seoane Pardo, A. M. (2018). Computational Thinking between Philosophy and STEM - Programming Decision Making Applied to the Behavior of “Moral Machines” in Ethical Values Classroom. *Revista Iberoamericana de Tecnologías Del Aprendizaje*, 13(1), 20–29. <https://doi.org/10.1109/RITA.2018.2809940>
- Strauss, A., Corbin, J. (1990) Basics of Qualitative Research: Grounded Theory Procedure and Techniques, Sage, Newbury Park, London.
- Tatar D., Harrison S., Stewart M., Frisina C., & Musaeus P. (2017) Proto-computational Thinking: The Uncomfortable Underpinnings. In: Rich P., Hodges C. (eds) *Emerging Research, Practice, and Policy on Computational Thinking. Educational Communications And Technology: Issues and Innovations*. Springer, Cham
- Thomas, J. O., Odemwingie, O. C., Saunders, Q., Watlerd, M., & Carlette Odemwingie, O. (2015). Understanding the Difficulties African-American Middle School Girls Face While Enacting Computational Algorithmic Thinking in the Context of Game Design. *Journal of Computer Science and Information Technology*, 3(1), 15–33. doi.org:10.15640/jcsit.v3n1a2
- University of Colorado (August 10, 2018). PhET Interactive Simulations. Retrieved from <https://phet.colorado.edu/>
- Vakil, S. (2014). A Critical Pedagogy Approach for Engaging Urban Youth in Mobile App Development in an After School Program. *Equity & Excellence in Education*. doi:10.1080/10665684.2014.866869
- Voogt, J., Eickelmann, B., Labusch, A., Yadav, A., Leftwich, A., Strijker, A., Rich, K., Good, J. & Sands, P. (2018). Learning and Teaching Computational Thinking – Challenges for Teacher Education. In E. Langran & J. Borup (Eds.), *Proceedings of Society for Information Technology & Teacher Education International Conference* (pp. 57-60). Washington, D.C., United States: Association for the Advancement of Computing in Education (AACE). Retrieved January 18, 2019 from <https://www.learntechlib.org/primary/p/182502/>
- 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. doi:10.1007/s10956-015-9581-5
- Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, 49(3), 33-35. doi:10.1145/1118178.1118215
- Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical transactions of the royal society of London A: mathematical, physical and engineering sciences*, 366(1881), 3717-3725. doi:10.1098/rsta.2008.0118
- Wing, J. (2010). *Computational thinking: What and why?* Carnegie Mellon School of Computer Science Discussion Papers. [Online] Available from: <https://www.cs.cmu.edu/>
- Wing, J. (2017). *A conversation about computational thinking*. Education: Future Frontiers. NSW Department of Education. Retrieved from https://education.nsw.gov.au/our-priorities/innovate-for-the-future/education-for-a-changing-world/media/documents/future-frontiers-education-for-an-ai-world/Computational-Conversation_1_A.pdf
- Wu, M. (2018). Educational Game Design as Gateway for Operationalizing Computational Thinking Skills among Middle School Students. *Ccsenet.Org*, 11(4), 15–28.

- doi:10.5539/ies.v11n4p15
- Yadav, A., Mayfield, C., Zhou, N., Hambrusch, S., & Korb, J. T. (2014). Computational Thinking in Elementary and Secondary Teacher Education. *ACM Trans. Comput. Educ.*, 14(16). doi:10.1145/2576872
- Yadav, A., Stephenson, C., & Hong, H. (2017). Computational thinking for teacher education. *Communications of the ACM*, 60(4), 55–62. doi:10.1145/2994591
- Yuen, T. T., & Robbins, K. A. (2015). A qualitative study of students' computational thinking skills in a data-driven computing class. *ACM Transactions on Computing Education (TOCE)*, 14(4), 27. doi:10.1145/2676660
- Zhong, B., Wang, Q., Chen, J., & Li, Y. (2016). An Exploration of Three-Dimensional Integrated Assessment for Computational Thinking. *Journal of Educational Computing Research*, 53(4), 562–590. doi:10.1177/0735633115608444
- Zur-Bargury, I., Parv, B., & Lanzberg, D. (2013). *A nationwide exam as a tool for improving a new curriculum*. Paper presented at the 18th ACM conference on Innovation and technology in computer science education, Canterbury, England, UK. doi: 10.1145/2462476.2462479