Article

Journal of Educational Computing Research
2022, Vol. 60(2) 481–511
© The Author(s) 2021
Article reuse guidelines:
sagepub.com/journals-permissions
DOI: 10.1177/07356331211033158
journals.sagepub.com/home/jec
SAGE

Computational Thinking Through an Empirical Lens: A Systematic Review of Literature

Ndudi O. Ezeamuzie and Jessica S. C. Leung

Abstract

This article provides an overview of the diverse ways in which computational thinking has been operationalised in the literature. Computational thinking has attracted much interest and debatably ranks in importance with the time-honoured literacy skills of reading, writing, and arithmetic. However, learning interventions in this subject have modelled computational thinking differently. We conducted a systematic review of 81 empirical studies to examine the nature, explicitness, and patterns of definitions of computational thinking. Data analysis revealed that most of the reviewed studies operationalised computational thinking as a composite of programming concepts and preferred definitions from assessment-based frameworks. On the other hand, a substantial number of the studies did not establish the meaning of computational thinking when theorising their interventions nor clearly distinguish between computational thinking and programming. Based on these findings, this article proposes a model of computational thinking that focuses on algorithmic solutions supported by programming concepts which advances the conceptual clarity between computational thinking and programming.

Keywords

computational thinking, problem solving, algorithms, programming, abstraction

Faculty of Education, University of Hong Kong, Hong Kong, Hong Kong

Corresponding Author:
Ndudi O. Ezeamuzie, Faculty of Education, University of Hong Kong, Hong Kong, Hong Kong.
Email: amuzie@connect.hku.hk

Introduction

In a well-cited seminal work on computational thinking (CT), Wing (2006) described CT as a fundamental problem-solving skill for everyone that is rooted in computer science concepts. Before Wing's pioneering call for CT, Alan Perils, the first recipient of the renowned A. M. Turing Award from the Association for Computing Machinery, had advocated for the inclusion of computer programming as part of liberal education for all students (Perils, 1962, as cited in Guzdial, 2008). A decade later, another renowned computer scientist, Donald Knuth, posited that people develop a clearer understanding of tasks by teaching a computer to perform them (Knuth, 1974). Similarly, in his work on procedural thinking, Seymour Papert promoted a vision of children teaching computers to think through programming (Papert, 1980). While Wing's conceptualisation has been explicitly acknowledged as the de facto reference for CT in the 21st century (e.g., Grover & Pea, 2013; Shute et al., 2017), the works of Perils, Knuth, and Papert embodied the spirit of CT and rightly underscored the association between computer science concepts and human cognition (Bull et al., 2020).

Wing's comparison of CT to the time-honoured literacies of reading, writing, and arithmetic was a brave claim. Subsequently, CT attracted interest from researchers, educators, and policymakers, leading to significant progress in the teaching and learning of CT (Hsu et al., 2018). One of the primary reasons for this spike in interest was the need to align education with the socio-digital revolution, which has changed the ways we study, work, and live. By leveraging CT skills, learners can design solutions for complex systems and instructors can teach the systems as they are (Buitrago Flórez et al., 2017). In this regard, CT encapsulates unique problem-solving skills to support non-reductionist and non-isolated approaches to learning –practices that are consistent with the objectives of 21st century science, technology, engineering, and mathematics (STEM) education.

While the vision of CT for everyone may be open to scholarly debate, the importance of CT is evidenced by a plethora of empirical findings. In their meta-analysis of a sample of 105 studies with 539 effect sizes, Scherer et al. (2019) discovered that computer programming, which is the predominant approach to developing students' CT skills (Lye & Koh, 2014), was positively associated with increase in students' cognitive skills. A similar meta-synthesis reported positive changes in programming knowledge when students learned CT through game programming (Denner et al., 2019). The pros of learning CT extend beyond programming knowledge to outcomes such as critical thinking, problem solving, collaboration, communication, and self-management (Popat & Starkey, 2019). Other advantages include positive attitudes and confidence (Denner et al., 2019). Therefore, it is not surprising that CT has been included in national curricula, such as the Next Generation Science Standards in the United States

(National Research Council, 2013) and curricula set by European ministries of education (Bocconi et al., 2016). It offers an invaluable means of tackling challenges facing 21st century education such as isolated subject learning, as demonstrated by the increased adoption of CT in formal curricula (Repenning et al., 2015). Even non-governmental organisations, such as the International Society for Technology in Education (ISTE), have recognised the importance of CT in their student standards (ISTE, 2016).

Nonetheless, the seemingly unbounded prospects offered by CT evoke equally critical questions about how it has been operationalized in empirical studies. What has been taught, learned, and assessed? These fundamental questions concerning the definition of CT have persisted. Indeed, Grover and Pea (2013) expressed similar concerns about the multiple definitions of CT. It is challenging to compare studies, standardise assessments, and generalise findings when faced with the multiple instructional strategies (see Buitrago Flórez et al., 2017; Hsu et al., 2018; Lye & Koh, 2014), diverse assessment methods (see Cutumisu et al., 2019; Shute et al., 2017; Tang et al., 2020), and ever-changing pedagogical tools (see Xia & Zhong, 2018; Yu & Roque, 2019; Zhang & Nouri, 2019) adopted in CT research. Furthermore, it is not clear whether the meaning of CT varies across educational settings, such as teachers' professional development (Menekse, 2015), higher education (Czerkawski & Lyman, 2015), and even in early childhood education (e.g., Bers, 2017).

Of course, we do not suggest that there is a shortage in CT conceptualisation. On the contrary, CT discourse is laden with diverse models that reflect varying perspectives. Such multifaceted definitions are not peculiar to CT but normal in educational research, where phenomena are examined through different lenses. Moreover, stepping back from the perceived challenges posed by a lack of consensus, the advantages of interpreting concepts through diverse lenses, including CT, cannot be ignored. Each definition contributes to a deeper understanding of the dimensions of CT. Standard conceptual models of CT found in studies include (a) the operational definition and age-appropriate examples produced by the International Society for Technology in Education and Computer Science Teachers Association (D. Barr et al., 2011; V. Barr & Stephenson, 2011), (b) the Computing At School CT framework (Csizmadia et al., 2015), (c) Brennan and Resnick's (2012) CT framework, (d) the CT framework for mathematics and science developed by Weintrop et al. (2016), and (e) the CT framework developed by Shute et al. (2017). These models reflect the near consensus that CT goes beyond programming, is not confined to computer science (Corradini et al., 2017; Lu & Fletcher, 2009; Nardelli, 2019), and is a transferable skill (Denning, 2017; Nardelli, 2019).

These models have supported CT research. In the current review, CT was interpreted according to Wing's (2006) theoretical proposition. Our choice was informed by the explicit distinction between CT and programming in Wing's (2006) model, which conceived CT primarily as a thinking style with or without programming a machine and rooted in computer science concepts. How have these computer science concepts been interpreted – in terms of their definition, consistency, and interaction across empirical studies? Certainly, any attempt to push for a consensus on CT would be an uphill task; we had no intention to engage in such a precarious venture. Nevertheless, a critical gap to be investigated relates to the ways in which CT has been operationalised in empirical studies, specifically with respect to the constituent elements. We expect a systematic investigation of these studies to reveal important patterns in how educators have interpreted CT. The findings of this review strengthen knowledge of CT and contribute to the evolving discourse. In addition, they offer valuable guidance for educators and policymakers on integrating CT into curricula and pedagogical practices.

Research Question: How has CT been defined in the literature investigating CT learning and assessment?

Shute et al. (2017) explored the nature of CT by examining the components, intervention strategies, and assessment methods. To the best of our knowledge, their quest to demystify CT is the closest to our study. However, our study is fundamentally different in two ways. (a) It considered not only the components of CT but also how these components are defined and interact with each other. (b) The findings of Shute et al. (2017) were based on a review of both empirical and theoretical works. In contrast, we aimed to identify how CT and its constituent elements have been operationalised in empirical interventions alone. In other words, we sought to ascertain, guided by our research question, how CT has been interpreted in practice.

Method

We adopted the Preferred Reporting Items for Systematic Reviews and Meta-Analyses (PRISMA) statement as the framing guideline. PRISMA was originally designed to provide a harmonised guideline for researchers conducting systematic reviews and meta-analyses in the field of medicine (Moher et al., 2009). However, this set of items also offers invaluable guidance for systematic reviews of research in other fields. Figure 1 summarises our use of PRISMA to identify and screen eligible studies.

Search Procedures

Our search parameters were influenced by the research question. We searched for articles that contained the keyword "computational thinking" in one or more of the following four fields: title, abstract, author keywords, and journal keywords. These fields represent significant parts of scholarly works that depict the central theme of an article. Note that the enclosing quotation marks were part of the search term; a schema for restricting the results to exact but

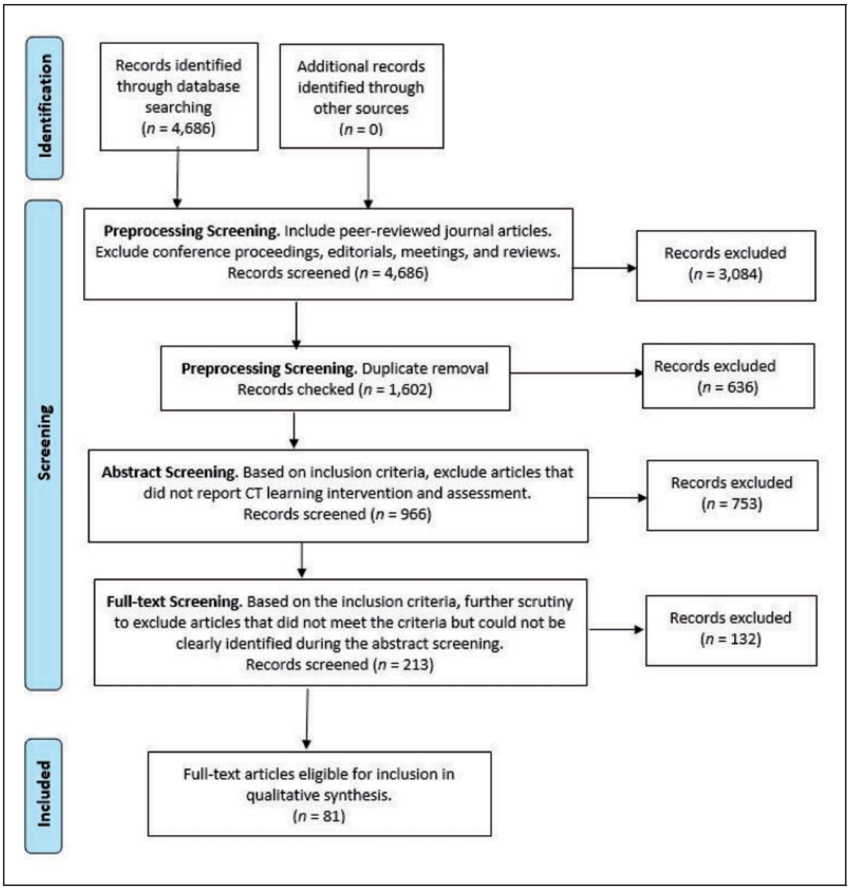
Identification
Records identified through database searching (n = 4,686)
Additional records identified through other sources (n = 0)
Screening
Preprocessing Screening. Include peer-reviewed journal articles. Exclude conference proceedings, editorials, meetings, and reviews. Records screened (n = 4,686)
Records excluded (n = 3,084)
Preprocessing Screening. Duplicate removal Records checked (n = 1,602)
Records excluded (n = 636)
Abstract Screening. Based on inclusion criteria, exclude articles that did not report CT learning intervention and assessment. Records screened (n = 966)
Records excluded (n = 753)
Full-text Screening. Based on the inclusion criteria, further scrutiny to exclude articles that did not meet the criteria but could not be clearly identified during the abstract screening. Records screened (n = 213)
Records excluded (n = 132)
Included
Full-text articles eligible for inclusion in qualitative synthesis. (n = 81)

Figure 1. Flow Diagram of Search Strategies and Screening Phases for Selecting Eligible Studies for Review.

case-insensitive matches. The same search term has been used in previous CT reviews (Hsu et al., 2018; Shute et al., 2017; Tang et al., 2020). While we found the methodological guidance paper on systematic review by Alexander (2020) invaluable in designing this study, the recommendation to use alternate terms for a major construct did not apply to our review, because the research question was formulated to explore studies with an explicit interest in CT intervention and assessment. Therefore, a single search term was an intuitive starting point that maximised coverage yet within the scope of our investigation.

With the search term identified, we executed our search in three key indexing databases for multidisciplinary educational research with comprehensive coverage of high-quality journal titles and proceedings: Web of Science (WoS), Scopus, and the Education Resources Information Center (ERIC). Scopus indexes works from more than 24,600 serial titles from over 5,000 publishers (Elsevier, n.d.) and WoS indexes the highest-quality research from more than 12,000 journals across 256 subject domains (Clarivate Analytics, n.d.). ERIC is a respected digital library of educational research that publishes innovative research for educators and the public.

Apart from restricting our search to the title, abstract, author keywords, and journal keywords, no further restrictions, such as the date of publication or language, were applied during this stage. Although our investigation was based on Wing (2006), we speculated that it was unnecessary to set such restrictions. In fact, any pre-2006 articles or written in languages other than English that evaded the screening might be useful to our investigation. The cut-off date of this identification stage was September 30, 2020, resulting in 4,686 documents: WoS ($n = 1{,}829$), Scopus ($n = 2{,}565$), and ERIC ($n = 292$). We considered extending our search results by journal scouring, referential backtracking, and researcher checking (Alexander, 2020). However, embarking on such procedures with 4,686 articles was not pragmatic at this stage.

Screening Procedures

With the initial pool of documents identified, we embarked on screening. Tracking back to the research question, documents were selected if they fulfilled the following two core inclusion criteria: (a) empirical studies with evidence of learners' engagement in a learning intervention and (b) studies that included assessments of learners' development of CT based on the intervention and not self-report measures.

These inclusion criteria mirrored our intention that studies that combined the learning and assessment of CT would provide clearer information on how the authors operationalised CT. The above inclusion criteria were designated 'core' because they were compulsory conditions for selecting documents that answered the research question. However, this also suggests that other inclusion and exclusion boundaries (e.g., peer-reviewed, source of document) were necessary to validate the quality of the selected documents and ensure a pragmatic review. Alexander (2020) referred to these boundaries as delimitations. The rationales for the delimitations are discussed in the following sections concerning the three sequential screening stages: pre-processing screening, abstract screening, and full-text screening.

Pre-Processing Stage. We decided to limit the pool of documents to peer-reviewed journal articles. Appropriate filters (for WoS and Scopus) were set to exclude other document types, including conference proceedings, editorials, meetings, and reviews. The decision to exclude the conference proceedings was based on their variable quality (Rothstein & Hopewell, 2009). Unfortunately, some articles that met the inclusion criteria might have been omitted by this delimitation. For example, an evaluation of teachers' development of CT practices (Kong & Lao, 2019), an examination of CT development when playing a game (Rowe et al., 2018), and an examination of the role of CT education in developing algorithmic thinking (Wong & Jiang, 2018) met the inclusion criteria. However, considering the pragmatism of reviewing more than 2,000 proceedings articles of variable quality, we excluded them. Moreover, concerns about publication bias when proceedings are excluded was diminished, since the question was not focused on effect size. The decision to exclude reviews and other document types (e.g. Lye & Koh, 2014; Repenning, 2012; Tissenbaum et al., 2019; Xia & Zhong, 2018) was relatively easy, because they did not meet the core inclusion criteria. We excluded 1,602 documents: 631 from WoS, 679 from Scopus, and 292 from ERIC. The final operation in the pre-processing stage was the removal of duplicates. The articles were aggregated, and duplicates ($n =$ 636) were removed. The remaining 966 articles underwent abstract screening.

Abstract Screening Stage. In this stage, we read the abstracts of the abovementioned 966 articles and screened out documents that failed to meet the two core inclusion criteria. We excluded reviews/meta-analyses (e.g. Scherer et al., 2019; Zhang & Nouri, 2019), theoretical discourse (e.g. Voogt et al., 2015; Yadav et al., 2016), studies of CT instrument development (e.g. Korkmaz et al., 2017; Román-González et al., 2017), studies of non-learning CT interventions (e.g. Israel et al., 2015), and studies that did not assess CT (e.g. Chang, 2014). The abstract screening culminated in the exclusion of 753 articles, leaving 213 articles for full-text screening.

Full-Text Screening Stage. The last screening procedure, full-text screening, provided a finer layer of inspection – an important process to assess the eligibility of articles when inclusion decisions could not be made based on their abstracts. Of the 213 remaining articles, 68 were excluded because they did not assess CT skills or focused substantially on programming. For example, Kalelioğlu (2015) investigated how teaching K–8 introductory computer science based on code.org influenced the development of students' reflective thinking skills towards problem solving. Although CT was mentioned approximately 12 times in Kalelioğlu's study, the full-text screening excluded the article because it used a self-reported scale. Similarly, other studies that measured related outcome variables but not CT skills were excluded. Examples included interest (e.g., Kong et al., 2018), self-efficacy (e.g., Psycharis & Kallia, 2017), and engagement (e.g., Liu et al., 2017). Another 16 studies were excluded because they used self-report measures to assess CT. Many of the articles in this category used the CT scale developed by Korkmaz et al. (2017) or variants of this scale. We excluded an additional 22 articles because their full texts were reported in languages other than English or were not available. We excluded 26 studies because they did not implement learning interventions or had unclear research designs.

Finally, after the full-text screening, we scoured the contents of journals that appeared more than once and looked at the publication records of authors who had contributed two or more studies. All the identified articles were found in our initial pool. This could be attributed to the increasing sophistication in database search algorithms and a single-term search keyword. The screening phase culminated in the inclusion of 81 eligible studies to be charted and analysed (see Supplementary Table).

Charting and Consolidating Data

After identifying eligible articles, we captured the information required to address our research question. Arguably, this was the most important stage of the study. It entailed several iterations of reading, analysis, and discussion of the 81 eligible articles by the research team. Due to the large number of articles designated for comprehensive review, we charted our data using a spreadsheet. The articles were catalogued to gain insight into their profiles and identify potential differences in CT operationalisation.

The captured variables were (a) authors, (b) title of article, (c) year of publication, (d) type of study (i.e. experimental, quasi-experimental, or exploratory), (e) number of participants, (f) duration of intervention, (g) research setting (i.e. in-class, after-school, or camp), (h) academic domain, (i) pedagogical approach, (j) nature of learning task (i.e. programming, non-programming, or both), (k) gender of participants (i.e. male, female, or mixed), (l) grade level of participants (i.e. kindergarten, lower elementary, upper elementary, middle school, high school, college), (m) nature of assessment task (i.e. programming, non-programming, or both), and (n) category of definition (see definitional coding section).

With respect to the research setting, studies conducted on school premises and within the regular class activities were coded as ‘in-class’. Studies conducted outside regular class hours such as extracurricular activities but on school premises were coded as ‘after-school’. When the studies were conducted during school holidays or outside the participants’ institution, the research setting was coded as ‘camp’.

To examine study type, articles were coded into three categories: experimental, quasi-experimental, or exploratory. Some studies reported interventions in which participants were split into separate groups (experimental versus control) and received different treatments. We coded the studies as experimental if the participants were randomly assigned to the groups; otherwise, the quasi-experimental label was assigned. The exploratory category comprised studies that investigated certain constructs or practices but did not establish experimental or control groups.

The nature of the learning and assessment tasks emerged as an important parameter. Although we had not set this variable at the outset, the iterative analysis flagged it as a construct of interest. We observed that learning and assessment tasks in the studies could be classified as either programming or non-programming tasks. If the participants wrote computer code as part of the learning activities (drag-and-drop or text), the nature of the learning task was coded as programming. Likewise, when writing computer code formed part of the assessment, the nature of the assessment was coded as programming.

Definitional Coding

Guided by the primary objective of our review – to investigate how CT has been defined, we were confronted with the challenge of making sense of a large number of studies. Dinsmore et al. (2008) encountered similar difficulty in their systematic review to disentangle the definitional boundaries between self-regulation, self-regulated learning, and metacognition. Informed by Murphy and Alexander's (2000) definitional coding scheme, which queried researchers' degree of explicitness in defining constructs, Dinsmore et al. (2008) coded definitions into two categories – implicit or explicit. They sub-categorised implicit definitions as conceptual definitions (i.e. the construct definition was inferred from words or phrases in the text), referential definitions (i.e. the construct definition was deduced from a referenced work), or measurement definitions (i.e. the construct definition was deduced from a measurement instrument).

In contrast, Singer and Alexander (2017) were interested in deconstructing the meaning of reading and opted to sub-categorise explicit definitions of reading as conceptual (what is reading?), componential (what does reading entail?), operational (how does reading occur?), or multifaceted (incorporating two or more conceptual, componential, and/or operational definitions). Based on these definitional coding schemes (Dinsmore et al., 2008; Murphy & Alexander, 2000; Singer & Alexander, 2017), we modified developed a three-tiered coding scheme for definitions of CT in the reviewed studies, focusing on the *explicitness*, *nature*, and *position* of definitions.

Explicitness was established if the researchers expressly stated the meaning or referenced the definition of CT. Clearly and deliberately stated definitions were coded as *explicit*, while inferential or implied definitions were coded as *implicit*. An additional code – *referential* – was assigned to the definitions when the final meanings were derived from other cited works.

The nature of the definitions was coded as *conceptual*, *componential*, or *multifaceted*. A definition was designated as conceptual if it captured the 'spirit' of CT by answering the question 'what is CT?' The componential code was assigned when CT was interpreted as a composite of other elements, constructs, concepts, or dimensions. The multifaceted code was assigned to CT definitions that incorporated both conceptual and componential elements.

Definition position – the last tier in our three-tiered model – was classified as *theoretical*, *intervention*, or *assessment*. An article was coded as theoretical when its definition of CT was identified from the framework or literature review that supported the study; as intervention when the definition was constructed from the learning or teaching experiment; and as assessment when the definition was deduced from the measurement instrument. In an ideal scenario, the definition in an article would be consistent across the three positions. As the positional values were not mutually exclusive, adopting a priority ranking (theoretical = 1, intervention = 2, and assessment = 3) was intuitive. We reasoned that definitions of the focal construct (CT) should ideally be established in the theoretical framework of an article. Our rationale for this priority ranking was connected to the general convention for reporting empirical studies of interventions. Researchers conceptualise the construct to be investigated, then implement an intervention based on the construct, and finally measure the effect of the intervention with instruments designed to assess the construct. Following the above ranking, the code 'theoretical' indicated that the definition was given in the theoretical background or literature review section of the article. However, this code did not indicate whether the definition was also provided in the intervention or assessment section. The code 'intervention' indicated that the definition was given in the section on implementing the intervention. While this suggested that the definition was not provided in the theoretical discussion, it did not indicate whether the definition was provided in the assessment section. The code 'assessment' implied that CT was defined in the assessment section only.

The following is a representative example of this three-tiered CT definitional coding. "To examine children's growing computational thinking ability throughout implementation of the TangibleK curriculum, four key variables were observed and assessed: debugging, correspondence, sequencing, and control flow" (Bers et al., 2014, p. 149). This was part of a theoretical discussion of the variables guiding CT investigation in early childhood education. We classified this as an explicit definition. Although there was no direct 'CT is . . .' phrase, the meaning of CT could be unambiguously decoded from the quotation, which was clearly aligned with the objective of querying how researchers operationalised CT in practice. In addition, CT was described as a composite construct with four dimensions (debugging, correspondence, sequencing, and control flow). Therefore, the nature of the definition was coded as componential. The position of the definition was best described as theoretical because it was part of the authors' theoretical background discussion of CT.

In terms of reliability, the coding and analysis process of the eligible articles involved several rounds of collaborative discussion and refinement. We considered this rigorous means of achieving a consensus on the coding as a proper reflection of reliability when multiple values were extracted from the articles.

Results and Discussion

Profiles of Charted Studies

Table 1 summarises the features of the studies ($n = 81$) in the final reviewed sample. Although our focus was on understanding how CT has been operationalised in empirical studies in the wake of Wing's (2006) call, our attention was immediately drawn to the absence of studies that implemented learning in tandem with the assessment of CT prior to 2013. Thereafter, the focus of CT research expanded. As we did not delimit our literature search by period, the paucity of studies that reported CT assessment before 2013 indicates that this period can be regarded as the infancy phase of CT development. At this juncture, our interest was drawn towards understanding the directions of research in the infancy stage as a precursor to making sense of trends in CT conceptualisation. Most of the empirical interventions reported before 2013 measured self-reported CT dispositions and other educational outcomes. Most of the studies in our initial pool were proceedings or theoretical works aimed at establishing the importance and boundaries of CT, such as rethinking approaches to teaching CT to everyone (Guzdial, 2008), redesigning CT instructional strategies (e.g. the three-phase 'use, modify, create' pedagogical framework proposed by Lee et al. (2011), representing the different cognitive levels of learning CT), and other plausible ways of introducing CT to students (e.g. Repenning's (2012) scalable game design).

We also noted the variety of instructional strategies deployed to facilitate the acquisition of CT skills. Effective teaching is an art that combines multiple pedagogical approaches to suit learners and learning content. Therefore, it was be difficult to infer the causative influence of all possible combinations of pedagogical approaches on the development of CT. However, the majority of the studies reported positive learning outcomes and highlighted the adaptability of CT to various teaching styles. Instructional strategies explicitly deployed in three or more studies are shown in Table 1. Other pedagogical approaches used to teach CT included feedback, portfolio design, peer-to-peer assessment, peer review, peer coaching, inquiry-based learning, project-based learning, debate, jigsaws and other puzzles, prototyping, mentoring, metaphor use, personalised learning, role play, role models, storytelling, and flipped learning.

With respect to gender, more than 90% of the studies were conducted with mixed-gender samples. There was only one study with male-only participants (Munoz et al., 2018) and one with female-only participants (Luo et al., 2020). Although we did not record the ratio of the male to female participants in the individual studies in our chart, which limits the ability to infer gender balance in CT, the deliberate effort to ensure equality is encouraging. Significant number of studies (77.78%, $n = 63$) were situated in school settings, indicating the positive characterisation of CT as a teachable skill in the classroom. However, a

Table 1. Summary of the Profiles of the Charted Studies.

Study characteristic	Value (*n*, %)
Year of publication	2013 (2, 2.47), 2014 (3, 3.70), 2015 (1, 1.23), 2016 (6, 7.41), 2017 (6, 7.41), 2018 (10, 12.35), 2019 (24, 29.63), 2020 (29, 35.80)
Type of study	Exploratory (50, 61.73), quasi-experimental (16, 19.75), experimental (15, 18.52)
Number of participants	1–10 (4, 4.94), 11–30 (11, 13.58), 31–100 (35, 43.21), 101–441 (30, 37.04), n/a (1, 1.23)
Duration of intervention (hours)	0–4.9 (12, 14.81), 5–9.9 (15, 18.52), 10–14.9 (11, 13.58), 15–24.9 (13, 16.05), 25–78 (11, 13.58), n/a (19, 23.46)
Research setting[a]	School (63, 77.78), camp (11, 13.58), after school (9, 11.11), n/a (3, 3.70)
Academic domain[a]	Technology (14, 17.28), science (10, 12.35), engineering (6, 7.41), mathematics (2, 2.47), others (4, 4.94), n/a (48, 59.26)
Instructional strategy[a]	Lecturing (39, 48.15), scaffolding (15, 18.52), collaboration (14, 17.28), learning-through-activities (10, 12.35), game-based (7, 8.64), problem-based (7, 8.64), simulation and modelling (7, 8.64), constructionism (6, 7.41), pair programming (5, 6.17), game-playing (4, 4.94), group work (4, 4.94), reflection (4, 4.94), constructivism (3, 3.70)
Gender	Female (1, 1.23), male (1, 1.23), mixed (74, 91.36), n/a (5, 6.17)
Grade level[a]	Kindergarten (4, 4.94), lower elementary (15, 18.52), upper elementary (30, 37.04), middle school (20, 24.69), high school (7, 8.64), college (15, 18.52), in-service teacher (3, 3.70), special needs (1, 1.23), n/a (3, 3.70)
Nature of learning tasks	Programming (62, 76.54), non-programming (17, 20.99), both (1, 1.23), n/a (1, 1.23)
Nature of assessment tasks	Non-programming (54, 66.67), programming (26, 32.10), both (1, 1.23)
Explicitness of definition	Explicit (57, 70.37), implicit (21, 25.93), n/a (3, 3.70)
Nature of definition	Componential (61, 75.31), conceptual (15, 18.52), multifaceted (2, 2.47), n/a (3, 3.70)
Position of definition	Assessment (28, 34.57), theoretical (27, 33.33), intervention (23, 28.40), n/a (3, 3.70)
Referential definition	Yes (39, 48.15), no (42, 51.85)

[a]The values for the study characteristics are not mutually exclusive.

sizeable number of studies (59.26%, $n = 48$) did not report how CT interfaced with academic domains or school subjects, which extends the conceptualisation of CT as a generic skill. The studies that reported how CT interfaced with academic domains were classified into the four domains of STEM: technology (17.28%, $n = 14$), science (12.35%, $n = 10$), engineering (7.41%, $n = 6$), and mathematics (2.47%, $n = 2$). Four studies explored CT acquisition outside the traditional STEM domains: psychology (Yadav et al., 2014), dance (Leonard et al., 2021), story-writing (Price & Price-Mohr, 2018), and professional development (Yadav et al., 2018). In summary, the development of CT was investigated from early childhood through college-level education. While the small number of investigations in kindergartens can be attributed to the probable cognitive load associated with learning CT, we were surprised by the paucity of interventions with high school students. A plausible explanation is the rigidity of curricula in high schools, where most students have concrete career directions in mind and computer science is an elective albeit well-established subject (Repenning et al., 2015).

Definition Explicitness

For clarity, the categorisation of definitions as explicit did not necessarily suggest that the authors presented their own definitions of CT or used definitional phrases such as 'CT is …'. Rather, it indicated that the meaning of CT was clearly articulated in the authors' descriptions of CT – whether it was conceived by the authors or extended from other references. For example, pointers such as "CT-test assesses the user's computational thinking level on five dimensions" (Taylor & Baek, 2019, p. 101) and "design for the programming for CT development curriculum was based on a framework adapted from Brennan and Resnick (2012)" (Kong et al., 2020, p. 5) were indicators of definitions were categorised as explicit.

Based on our coding, we found that 70.37% of the studies ($n = 57$) offered explicit definitions. Implicit definitions were found in 25.93% of the studies ($n = 21$). The definitions in the remaining 3.70% of the studies ($n = 3$) were unclear and designated as undefined. All the implicit and undefined categories were logged as non-referential definitions because the meanings and absence of CT respectively, were deduced directly from the primary studies.

Of the studies that explicitly defined CT ($n = 57$), 68.42% ($n = 39$) gave referential definitions, in contrast with 31.58% ($n = 18$) that were coded as non-referential definitions. This indicated that more than two thirds of the studies that explicitly defined CT, adduced its operational meaning from existing CT frameworks. While this finding may be encouraging and aligned with the golden rule of hinging educational research on sound theory, other explanations are possible. As we demonstrate in a later section, this trend may also be linked to the blurry understanding of CT and/or preference for using existing measurement instruments.

Explicit and Conceptual Definitions. Surprisingly, only four articles provided explicit conceptual/multifaceted definitions (Allsop, 2019; Kim et al., 2013; Rodríguez del Rey et al., 2021; Yadav et al., 2018). Kim et al. (2013) provided an explicit conceptual definition, describing CT as "a sort of logical thinking every human being has" (p. 443) in an experimental investigation of the effectiveness of paper-and-pencil programming over Logo in preservice teachers' classroom. They measured the CT skills acquired by the preservice teachers using the Group Assessment of Logical Thinking, an instrument designed primarily to measure logical reasoning skills. This depicts that the authors inferred a strong association between CT and logical thinking. Another explicit conceptual definition was reported in a study of a professional development programme. Yadav et al. (2018) defined CT as "the ways of thinking, or habits of mind, that computer scientists use" (p. 379), which is consistent with Wing's (2006) seminal interpretation of CT. This broad definition was crafted to lead in-service teachers to integrate CT with mathematics and science, reflecting on the meaning. Through vignette challenges, Yadav et al. (2018) deduced that CT *is* programming and problem-solving; that it *involves* the use of logic, algorithms, data manipulation, and pattern recognition; and that it *aids* in prediction and improving efficiency.

The other two explicit definitions were most appropriately classified as multifaceted – a blend of conceptual and componential definitions. Allsop (2019) described CT as a cognitive process regulated by metacognitive practices with the aim of using computational concepts (i.e. sequences, loops, events, parallelism, conditionals, operators, variables, abstraction) to automate solutions to problems. According to Rodríguez del Rey et al. (2021), "CT is a cognitive process executed by humans for the resolution of diverse problems using computational concepts" (p. 3). While both multifaceted definitions highlighted the cognitive aspect of CT, which is consistent with the assumption of a relationship between CT and logical thinking (Kim et al., 2013), we observed disparities in their underlying CT concepts. Besides the overlapping abstraction, Rodríguez del Rey et al. (2021) implemented CT concepts differently, focusing on data processing, decomposition, algorithms, generalisation, simulation, and evaluation.

Explicit and Componential Definitions. Table 2 summarises the explicit componential definitions of CT, which did not originate in another research. These definitions expressed how the researchers operationalised CT in their studies without wholly adopting pre-existing frameworks. Of course, this does not suggest that the authors failed to situate their work in relation to existing frameworks. Rather, these definitions reflected the meanings that the authors assigned to CT based on their review of the literature. Although CT components varied across

Table 2. Non-Referential, Explicit, and Componential Conceptualisations of Computational Thinking (CT).

Author(s)	Definition
Bers et al. (2014)	CT variables – debugging, sequences, correspondence, flow control
Yadav et al. (2014)	CT concepts – problem identification, decomposition, abstraction, logical thinking, algorithms, debugging
Atmatzidou & Demetriadis (2016)	CT dimensions – abstraction, generalisation, algorithms, decomposition, modularity
Atmatzidou & Demetriadis (2017)	CT concepts – abstraction, generalisation, algorithms, decomposition, modularity, debugging
Looi et al. (2018)	CT skills – decomposition, algorithms, abstraction, generalisation, evaluation
Witherspoon et al. (2018)	CT concepts – sequences, conditionals, iteration
Tran (2019)	CT concepts – sequences, algorithms, looping, debugging, conditionals
Nam et al. (2019)	Forms of CT – sequencing, problem solving
Calderon et al. (2020)	CT elements – abstraction, decomposition, data, algorithms, sequences
Chen et al. (2020)	CT items – creativity, valuableness, simplification, embedding, simulation, transformation
Angeli & Valanides (2020)	CT elements – algorithm, sequencing, decomposition, debugging
Noh & Lee (2020)	CT components – data collection, data analysis, structuring, decomposition, modelling, algorithm, automation, generalisation
Yin et al. (2020)	CT subskills – decomposition, abstraction, algorithm, pattern generalisation
Uzumcu & Bay (2020)	CT dimensions – problem understanding, flowchart, operators, conditionals, loops, parallelism, decomposition, abstraction, pattern, algorithms, evaluation, debugging

the studies, several CT concepts (also referred to as dimensions, elements, variables, skills, items, and subskills) overlapped. As reported in Table 2, while there was no consensus on a definition of CT, the factors considered were highly consistent, signifying that most of the CT components identified in these studies were meaningfully aligned with computer science concepts and practices. They were also consistent with sets of components identified in previous review articles, such as the five components (abstraction, decomposition, algorithms, evaluation, generalisation) in Selby and Woollard (2013) and six components (decomposition, abstraction, algorithms, debugging, iteration, generalisation) in Shute et al. (2017). Conceptual consistency denoted alignment with computer science concepts and practices; not to be misconstrued or translated into accuracy of implementation in the studies. A notable exception to the conceptual consistency was a study that explored the effects of an identification, modelling, simulation, and prototyping teaching model in a college design course (Chen et al., 2020). CT was operationalised in the assessment as a composite value reflecting creativity, valuableness, simplification, embedding, simulation, and transformation.

Implicit and Componential Definitions. Besides the explicit componential definitions in Table 2, we deduced implicit componential definitions in eight studies. For example, in a study investigating the relationship between spatial reasoning and CT with primary school students (Città et al., 2019), CT was not explicitly framed. However, the CT test measured the students' ability to write and interpret algorithms on paper and CT was implicitly coded as a composite of sequences and algorithms.

Similarly, decomposition, abstraction, logic, and algorithms were implicit components of CT in Price and Price-Mohr's (2018) study. Although it is desirable to explicitly define constructs in educational research, the implicit components of CT were generally consistent with the explicit components (see Table 2). An exception to the consistency was González-González et al. (2019), who included software and hardware amongst the components of CT. While these were unconventional corpora of computer science concepts that are associated with CT, it appears that the authors included these components to make the learning intervention relatable to students with Down syndrome.

Implicit and Conceptual Definitions. The articles in this category ($n = 13$) asserted that participants engaged in CT learning interventions and were assessed afterwards. However, we could not deduce any explicit definitions or implied components of CT from the articles. The teaching and learning of programming represented the common trend in these studies, such as the use of Python programming language to teach problem-solving and biological computational techniques in a college bioscience course (Libeskind-Hadas & Bush, 2013). Therefore, articles that focused on computer programming as a measure of CT without an obvious meaning of CT were designated as offering implicit conceptual definitions.

Basogain et al. (2018) reported the use of learning strategies such as blended learning, constructionism, continuous assessment, feedback, portfolio creation, and peer-to-peer assessment for CT study. As part of the learning and assessment, the participants wrote programs in a visual drag-and-drop environment and documented their learning in portfolios. Unfortunately, the features of the rubric used in the peer-to-peer assessment were not described. Other related studies compared how programming via virtual and physical robotic platforms supported CT development in middle school (Berland & Wilensky, 2015) and investigated the roles of bodily engagement and computational perspectives in the learning of CT with a code-and-play application in primary school (Sung & Black, 2020).

Although the authors' intention in the studies in this category was to explore CT, they focused on programming. While programming may be a viable approach to learning CT (Lye & Koh, 2014), this focus blurred the boundary between programming and CT.

Referential Definitions

Building on existing theories remains a cherished practice in research. By situating empirical investigations on sound frameworks, researchers add to the pool of knowledge in systematic and meaningful ways. As stated earlier, the majority (68.42%, $n = 39$) of articles within the explicitly defined category extracted CT definitions from previous studies or frameworks. Of the 39 articles, 35.90% ($n = 14$) gave these definitions in the theory section. On the other hand, 10.26% ($n = 4$) extracted their definitions from the implemented intervention, while a remarkable 53.85% ($n = 21$) definitions were deduced from the assessment.

Table 3 lists the frameworks cited by the articles in the final pool. Brennan and Resnick's (2012) framework was the most frequently cited. They framed CT in three dimensions: concepts (i.e. the knowledge that designers have and apply when they program), practices (i.e. what designers do when programming), and perspectives (i.e., how designers see themselves and their environments). This componential definition has been used to frame CT in diverse scenarios. For example, it was adapted to design programming activities for teacher development programmes in primary schools (Kong et al., 2020) and to understand differences by gender in eye tracking when programming (Papavlasopoulou et al., 2020). The prevalence of Brennan and Resnick's (2012) framework may not be unconnected with the wide adoption of Scratch. Scratch is a free block-based visual programming platform for children, designed in compliance with the 'low threshold, high ceiling' requirement for CT tools (Repenning et al., 2010) and has several million users. The framework and scratch are linked outputs of a design-based learning research at the Massachusetts Institute of Technology.

As well as framing empirical studies, Brennan and Resnick's (2012) framework offers a reliable model for interpreting CT in review articles (e.g. Lye & Koh, 2014; Zhang & Nouri, 2019). Underpinned by this framework, Lye and Koh (2014) reviewed 27 empirical studies to unmask the connections between CT and programming in K–12 education. Zhang and Nouri (2019) used the framework to identify CT skills learned via K–9 Scratch in their systematic review of 55 empirical studies. These researchers found that all of the components of Brennan and Resnick's (2012) framework were successfully captured in

Table 3. Definition of Computational Thinking From the Referenced Literature.

Referenced article	Referencing articles (*n*)	Framing of computational thinking
Brennan & Resnick (2012)	14	CT concepts – sequences, loops, parallelism, events, conditionals, operators, and data CT practices – incremental and iterative design, testing and debugging, reusing and remixing, and abstracting and modularising CT perspectives – expressing, connecting, and questioning
Moreno-León et al. (2015)	8	Abstraction and problem decomposition, logical thinking, synchronisation, parallelism, algorithmic flow control, user interactivity, and data representation. Also referred to as Dr. Scratch.
Román-González et al. (2017)	6	Sequences, loops, conditionals, functions, and variables. Also referred to as Computational Thinking Test (CTt).
Dagienė & Sentance (2016)	4	Abstraction, algorithms, decomposition, evaluation, and generalisation. Also referred to as Bebras task.
D. Barr et al. (2011), ISTE & CSTA, (2011)	2	Problem solving that incorporates the characteristics of problem formulation, abstraction, logical thinking, algorithms, efficiency, generalising, and transfer
Csizmadia et al. (2015), Selby & Woollard (2013)	2	Abstraction, decomposition, algorithms, evaluation, and generalisation
Weintrop et al. (2016)	1	Data practice, simulation and modelling, problem solving, system thinking
CMCCT (n.d.)	1	Abstraction, algorithms

Note. CMCCT = Carnegie Mellon Center for Computational Thinking, CSTA = Computer Science Teachers Association, ISTE = International Society for Technology in Education.

the reviewed empirical studies. While CT concepts were predominantly explored in the literature, CT perspectives were the least studied. Zhang and Nouri (2019) identified additional skills in the literature missing from Brennan and Resnick's (2012) framework. They suggested revision of the framework that would add input and output to CT concepts; user interaction to CT perspectives; and multimodal design, predictive thinking, and the ability to read, interpret, and communicate code to the CT practice dimension. Whether to revise or not, these suggestions revealed that there may still be unknown gaps in the framing of CT.

One article (Newton et al., 2020) referenced Repenning et al. (2015) as the framework for analysing CT in students' games but assessed abstraction, algorithms, and learning transfer as factors of CT. The authors presented these factors as the summative categories from the CT pattern of Repenning et al. (2015). Other frameworks cited included Dr. Scratch (Moreno-León et al., 2015), CT test (Román-González et al., 2017), and Bebras (Dagienė & Sentance, 2016). These three models were used in 46.15% of the articles that defined CT referentially. Collectively, their designs incorporated measurement instruments for CT, especially Dr. Scratch, which automatically analyses Scratch projects. A substantial number of articles adopted these frameworks, along with the 53.85% of the articles that defined CT in the assessment section. This supports our initial observation that referential definitions may be linked to the blurry understanding of CT and/or preference for using existing measurement instruments.

Components of Computational Thinking

Of the reviewed articles, 75.31% ($n = 61$) interpreted CT as a collection of constructs and components. This interpretation was shared by most of the researchers but contrasted with the definitions in the remaining articles – 13 conceptually framed CT as programming, 4 offered explicit conceptual definitions of CT, and 3 did not define CT. As most of the researchers operationalised CT as a composite of components, we were persuaded to query which components were associated with CT.

Based on a significance level of 0.05, we classified components that appeared in more than 5% of the articles as significant. In contrast, non-significant components were those that appeared in fewer than 5% of the articles. Table 4 shows the aggregated frequency of the observed components and Figure 2 is a simple word cloud visualisation of the components of CT. In a word cloud, the font weight of a word is directly proportional to its frequency.

The components identified as significant, such as abstraction, sequences, algorithms, decomposition, and debugging, are prominent dimensions of common CT models (e.g. Brennan & Resnick, 2012; ISTE & CSTA, 2011; Shute et al., 2017). Although the significant components seemed to overlap with the existing models, the sizeable number of non-significant components explains the challenges in delineating a CT model. Given that computer science concepts are infinite, variations in the components and definitions of CT are likely to persist.

The Boundary Between Computational Thinking and Programming

An overwhelming 77.78% ($n = 63$) of the interventions were based on the learning of programming. Moreover, 33.33% ($n = 27$) of the studies assessed learners'

Table 4. Significant vs. Non-significant Components of Computational Thinking in the Literature.

Significant components (*n*)	Non-significant components
Abstraction (38), sequence (31), conditional (26), algorithm (24), loop (23), parallelism (23), debugging (22), data (20), decomposition (16), event (15), iteration (15), modularity (14), testing (13), logic (12), operator (12), remixing (12), reusing (12), generalisation (11), flow control (9), evaluation (8), synchronisation (8), user interactivity (8), variable (5), function (4)	Transfer, design, efficiency, modelling, pattern, problem formulation, problem solving, simulation, system thinking, app originality, automation, causal inference, clear instruction, control, correspondence, creativity, data analysis, data collection, develop, embedding, experimentation, expressions, extraneous block, flowchart, functionality, hardware, instruction, keyboard input, list, object, operators, planning, problem identification, problem understanding, programming, random number, representation, robot programming, simplification, software, sprite customisation, sprite name, stage customisation, structuring, transformation, use, user interface, valuableness

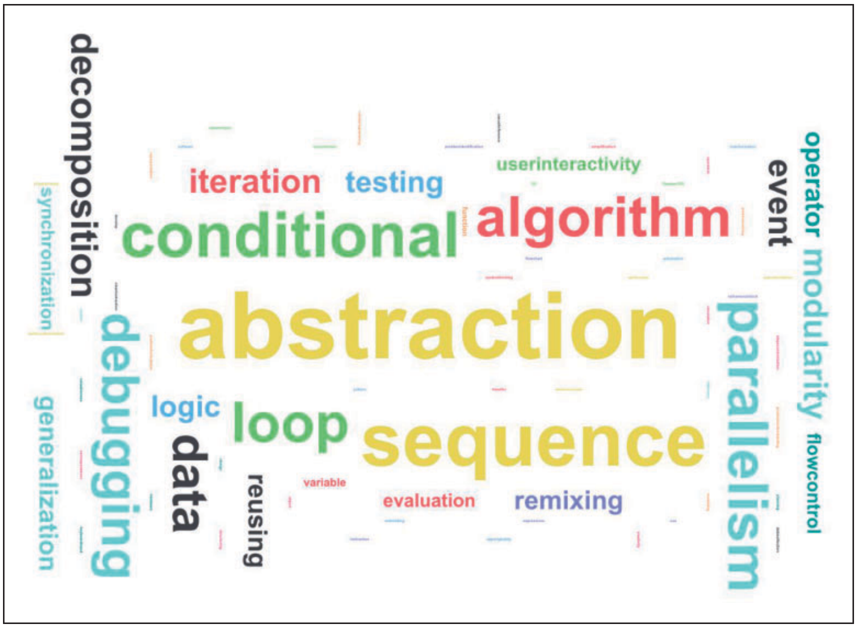

Figure 2. Word Cloud Representation of the Components of Computational Thinking.

programming artefacts. Examples include the acquisition of CT through programming with Scratch in an extracurricular computing programme that investigated the effects of repeated participation (Mouza et al., 2020), the learning of CT through Python programming in an undergraduate bioscience course (Libeskind-Hadas & Bush, 2013), and the programming of robots as part of the development of a CT instrument (Chen et al., 2017). The above finding is consistent with the observation that programming is the main method of teaching and learning CT (Lye & Koh, 2014). However, the distinction between CT and programming was narrow in these studies, especially when the most prominent feature of CT is "conceptualizing, not programming" (Wing, 2006, p. 35).

There is a near consensus among researchers that CT is a thinking style for solving problems with or without programming. We discovered that the meaning of CT in the reviewed articles' theoretical discussions often did not match the operationalised meaning. Conceptually, the researchers understood that CT is not equivalent to programming. However, several studies assessed CT through programming artefacts. Some studies that did not require learners to write computer programs nevertheless assessed learners' knowledge of programming concepts as a measure of CT through tests.

Understanding what Wing (2006) meant by "drawing on the concepts fundamental to computer science" (p. 33) illuminates the overlap between CT and programming. Computer science is overly broad. Its branches include but are not limited to databases, networking, software engineering, graphics, artificial intelligence, virtual reality, cryptography, and computer security (V. Barr & Stephenson, 2011). Professionals in different branches of computer science require different skill sets. For example, the knowledge and skills required for the effective practice of cryptography versus artificial intelligence are different. Nevertheless, the different branches of computer science require some common skills, such as programming, from practitioners. Indeed, programming is an indisputable learning module and essential skill for every computer scientist.

Considering the diverse nature of computer science, only a subset of the concepts and practices in this domain may qualify as components of CT. Otherwise, various computer science concepts that did not appear in our analysis of componential definitions (e.g. inheritance, encapsulation, and polymorphism), could justifiably be included as components of CT in future studies. Table 4 shows the components of CT that were extracted from the reviewed empirical studies. In our view, these components closely reflect the programming concepts that computer scientists use to solve problems, irrespective of the specific domain. Therefore, we deduce that these programming concepts are most appropriate for describing CT. This also explains the close association between CT and programming. By 'programming', we mean not only writing code (coding) but also the series of activities involved in software engineering, from ideation to production and from problem statement to problem solution.

Summarising the Computational Thinking Frameworks

Here, we summarise what the systematic review of the literature taught us about definitions and operationalisation of CT. Based on these findings, we adduce a definition to explain and frame the nature of CT. This definition was deduced from two principles on which researchers have generally agreed. The first principle is that CT is a skill underpinned by computer science concepts. Concisely, programming concepts are the underlying support for CT. The second principle positions CT as a relevant cognitive skill that is transferable to problem solving in other domains. We emphasise 'other domains' because this differentiates CT from programming. Otherwise, CT may not be much more than a reinvention of programming. Undoubtedly, programming is a useful skill and will remain a core method of acquiring CT. However, the ability to use this skill for everyday problem solving distinguishes CT from programming.

Therefore, we understood CT as the cognitive skill required to design algorithmic solutions for problems in different knowledge areas. By 'algorithmic solution', we refer to the "series of steps that control some abstract machine or computational model without requiring human judgement" (Denning, 2017, p. 33). This helps to distinguish an algorithm from just any sequence of steps.

What constitutes programming concepts varies greatly (see Table 4) and cannot be conclusively bound in a finite list. Moreover, some concepts can be operationalised in diverse ways. For example, abstraction is a key concept in programming (Kramer, 2007), generally understood as the practice of focusing on major details while concealing the peripherals. However, abstraction occurs with different degrees of granularity, and practices such as modularisation, problem decomposition, pattern generation, and problem formulation are all forms of abstraction. Therefore, our definition emphasises the development of algorithmic solutions whereby learners draw on their knowledge of programming concepts in non-isolated approach.

Conclusions and Implications

We embarked on this review with the goal of identifying ambiguities and gaining clarity in the operationalisation of CT; a 21st century skill for navigating daily procedural tasks efficiently and a cognitive ability transferable to problem solving in other domains (Denning, 2017). How have researchers operationalised CT in empirical studies? In our own practice, we have long faced the challenge of distinguishing between CT and computer programming. Indeed, this is not peculiar particular and has often been acknowledged in the CT literature. For example, consider how Nardelli (2019) described Wing's (2006) seminal article as a call to "start rolling the ball" towards the introduction of computer science and informatics for all school students (p. 33). While acknowledging Nardelli's (2019) argument that it may be risky to emphasise a few aspects of Wing's (2006) model and promote CT as a distinct subject, alternative interpretations – such as the idea that CT is for everyone, not only computer scientists – cannot be overlooked when programming and computer science are distinct subjects with considerable high cognitive demand.

To highlight the educational implications of this review, we must acknowledge its limitations and delimitations. The major constraint we established was the exclusion of studies that did not address the learning and assessment of CT. We also restricted our search to peer-reviewed articles. Inevitably, these delimitations excluded reviews and theoretical literature. These decisions led to the exclusion of some articles that may have offered additional insights into the subject. For example, Zhong et al. (2016) investigated how pair programming aided in the development of CT, but their study was excluded because it did not measure CT. However, our decision to impose these delimitations was strongly tied to our core inquiry into the definitions and operationalisation of CT in empirical studies. Therefore, without diminishing the roles of reviews and theoretical articles in framing this exercise, they were justifiably excluded from the main investigation.

Although we positioned our review based on Wing (2006), Papert's (1980) constructionist approach suggests that children can develop a special reasoning style through the epistemic practice of teaching computers to think, which can increase children's confidence in solving problems with different cognitive styles. In our interpretation, teaching computers to think is analogous to computer programming, and the reasoning style described by Papert (1980) is equivalent to CT as described by Wing (2006). CT derives its popularity from the consensus that it is not limited to programming but a cognitive skill for everyone that is transferable to other learning domains. However, most studies in this area have concentrated on programming. Although it is indisputable that a strong association exists between CT and programming, future research should explore other ways in which CT can be operationalised and transferred to other domains.

Most of the reviewed studies framed CT as a combination of concepts, such as abstraction, algorithms, decomposition, and sequences. While the components varied considerably, they were aligned with the concepts that computer scientists use when writing computer programs to solve a problem. In future research, it would be desirable to investigate whether the components of CT are consistently operationalised both within and between studies. Also, exploring the correlation of the different operationalization of CT and educational outcomes will deepen understanding of CT. Considering the sizeable number of definitions deduced from the interventions or assessment instruments, researchers will add clarity to CT operationalisation by explicitly stating their theoretical position.

Our overarching aim in seeking definitional clarity was to enhance the teaching, learning, and assessment of CT. As stated earlier, our objective was not to seek a consensus on a definition of CT. In fact, based on our analysis of the trends in componential definitions, there is no sign of convergence in the literature. We propose a definition that focuses on algorithmic problem solving, supported by various programming concepts. Within this framing, CT can be effectively integrated with learning effectively without focusing on individual concepts in isolation. Thereby, researchers and instructors can avoid common pitfalls in conceptualising CT.

Declaration of Conflicting Interests

The authors declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The authors received no financial support for the research, authorship, and/or publication of this article.

ORCID iDs

Ndudi O. Ezeamuzie https://orcid.org/0000-0001-8946-5709
Jessica S. C. Leung https://orcid.org/0000-0002-6299-8158

Supplemental material

Supplemental material for this article is available online.

References

Alexander, P. A. (2020). Methodological guidance paper: The art and science of quality systematic reviews. *Review of Educational Research*, *90*(1), 6–23. https://doi.org/10.3102/0034654319854352

Allsop, Y. (2019). Assessing computational thinking process using a multiple evaluation approach. *International Journal of Child-Computer Interaction*, *19*, 30–55. https://doi.org/10.1016/j.ijcci.2018.10.004

Angeli, C., & Valanides, N. (2020). Developing young children's computational thinking with educational robotics: An interaction effect between gender and scaffolding strategy. *Computers in Human Behavior*, *105*, 105954. https://doi.org/10.1016/j.chb.2019.03.018

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.

Atmatzidou, S., & Demetriadis, S. (2017). A didactical model for educational robotics activities: A study on improving skills through strong or minimal guidance. In: D. Alimisis, M. Moro, & E. Menegatti (Eds.), *Educational robotics in the makers era* (pp. 58–72). Springer.

Barr, D., Harrison, J., & Conery, L. (2011). Computational thinking: A digital age skill for everyone. *Learning Leading with Technology*, *38*(6), 20–23.

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.

Basogain, X., Olabe, M. Á., Olabe, J. C., & Rico, M. J. (2018). Computational thinking in pre-university blended learning classrooms. *Computers in Human Behavior*, *80*, 412–419. https://doi.org/10.1016/j.chb.2017.04.058

Berland, M., & Wilensky, U. (2015). Comparing virtual and physical robotics environments for supporting complex systems and computational thinking. *Journal of Science Education and Technology*, *24*(5), 628–647. https://doi.org/10.1007/s10956-015-9552-x

Bers, M. U. (2017). *Coding as a playground: Programming and computational thinking in the early childhood classroom*. Routledge.

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*(C), 145–157.

Bocconi, S., Chioccariello, A., Dettori, G., Ferrari, A., & Engelhardt, K. (2016). *Developing computational thinking in compulsory education—Implications for policy and practice*. Publications Office of the European Union. https://doi.org/10.2791/792158

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

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.

Bull, G., Garofalo, J., & Hguyen, N. R. (2020). Thinking about computational thinking. *Journal of Digital Learning in Teacher Education*, *36*(1), 6–18. https://doi.org/10.1080/21532974.2019.1694381

Calderon, A. C., Skillicorn, D., Watt, A., & Perham, N. (2020). A double dissociative study into the effectiveness of computational thinking. *Education and Information Technologies*, *25*(2), 1181–1192. https://doi.org/10.1007/s10639-019-09991-3

Carnegie Mellon Center for Computational Thinking. (n.d.). *What is computational thinking*? https://www.cs.cmu.edu/~CompThink/index.html

Chang, C.-K. (2014). Effects of using Alice and scratch in an introductory programming course for corrective instruction. *Journal of Educational Computing Research*, *51*(2), 185–204. https://doi.org/10.2190/EC.51.2.c

Chen, G., He, Y., & Yang, T. (2020). An ISMP approach for promoting design innovation capability and its interaction with personal characters. *IEEE Access*, *8*, 161304–161316. https://doi.org/10.1109/ACCESS.2020.3019290

Chen, G., Shen, J., Barth-Cohen, L., Jiang, S., Huang, X., & Eltoukhy, M. (2017). Assessing elementary students' computational thinking in everyday reasoning and robotics programming. *Computers & Education*, *109*, 162–175. https://doi.org/10.1016/j.compedu.2017.03.001

Città, G., Gentile, M., Allegra, M., Arrigo, M., Conti, D., Ottaviano, S., Reale, F., & Sciortino, M. (2019). The effects of mental rotation on computational thinking. *Computers & Education*, *141*, 103613. https://doi.org/10.1016/j.compedu.2019.103613

Clarivate Analytics. (n.d.). *Web of science core collection*. https://clarivate.com/webofsciencegroup/solutions/web-of-science/

Corradini, I., Lodi, M., & Nardelli, E. (2017). Conceptions and misconceptions about computational thinking among Italian primary school teachers. In: J. Tenenberg, D. Chinn, J. Sheard, & L. Malmi (Eds.), *Proceedings of the 2017 ACM conference on international computing education research* (pp. 136–144). ACM. https://doi.org/10.1145/3105726.3106194

Csizmadia, A., Curzon, P., Dorling, M., Humphreys, S., Ng, T., Selby, C., & Woollard, J. (2015). Computational thinking: A guide for teachers. https://eprints.soton.ac.uk/424545/

Cutumisu, M., Adams, C., & Chang, L. (2019). A scoping review of empirical research on recent computational thinking assessments. *Journal of Science Education and Technology*, *28*(6), 651–676. https://doi.org/10.1007/s10956-019-09799-3

Czerkawski, B. C., & Lyman, E. W. (2015). Exploring issues about computational thinking in higher education. *TechTrends*, *59*(2), 57–65. https://doi.org/10.1007/s11528-015-0840-3

Dagienė, V., & Sentance, S. (2016). It's computational thinking! Bebras tasks in the curriculum. In: A. Brodnik & F. Tort (Eds.), *Informatics in schools: Improvement of informatics knowledge and perception* (pp. 28–39). Springer. https://doi.org/10.1007/978-3-319-46747-4_3

Denner, J., Campe, S., & Werner, L. (2019). Does computer game design and programming benefit children? A meta-synthesis of research. *ACM Transactions on Computing Education*, *19*(3), 1–35. https://doi.org/10.1145/3277565

Denning, P. J. (2017). Remaining trouble spots with computational thinking. *Communications of the ACM*, *60*(6), 33–39. https://doi.org/10.1145/2998438

Dinsmore, D. L., Alexander, P. A., & Loughlin, S. M. (2008). Focusing the conceptual lens on metacognition, self-regulation, and self-regulated learning. *Educational Psychology Review*, *20*(4), 391–409. https://doi.org/10.1007/s10648-008-9083-6

Elsevier. (n.d.). *Getting the most out of published research*. https://www.elsevier.com/solutions/scopus/how-scopus-works

González-González, C. S., Herrera-González, E., Moreno-Ruiz, L, Reyes-Alonso, N., Hernández-Morales, S., Guzmán-Franco, M. D., & Infante-Moro, A. (2019). Computational thinking and down syndrome: An exploratory study using the KIBO robot. *Informatics*, *6*(2), 25. https://doi.org/10.3390/informatics6020025

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

Guzdial, M. (2008). Paving the way for computational thinking. *Communications of the ACM*, *51*(8), 25–27. https://doi.org/10.1145/1378704.1378713

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

International Society for Technology in Education. (2016). *ISTE standards for students*. https://www.iste.org/standards/for-students

International Society for Technology in Education, & Computer Science Teachers Association. (2011). *Operational definition of computational thinking for K-12 education*. http://www.iste.org/docs/ct-documents/computational-thinking-operational-definition-flyer.pdf?sfvrsn = 2

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

Kalelioğlu, F. (2015). A new way of teaching programming skills to K-12 students: Code.org. *Computers in Human Behavior*, *52*, 200–210. https://doi.org/10.1016/j.chb.2015.05.047

Kim, B., Kim, T., & Kim, J. (2013). Paper-and-pencil programming strategy toward computational thinking for non-majors: Design your solution. *Journal of Educational Computing Research*, *49*(4), 437–459. https://doi.org/10.2190/EC.49.4.b

Knuth, D. E. (1974). Computer science and its relation to mathematics. *The American Mathematical Monthly*, *81*(4), 323–343. https://doi.org/10.1080/00029890.1974.11993556

Kong, S.-C., Chiu, M. M., & Lai, M. (2018). A study of primary school students' interest, collaboration attitude, and programming empowerment in computational thinking education. *Computers & Education*, *127*, 178–189. https://doi.org/10.1016/j.compedu.2018.08.026

Kong, S.-C., Lai, M., & Sun, D. (2020). Teacher development in computational thinking: Design and learning outcomes of programming concepts, practices and pedagogy. *Computers & Education*, *151*, 103872. https://doi.org/10.1016/j.compedu.2020.103872

Kong, S.-C., & Lao, A. C.-C. (2019). Assessing in-service teachers' development of computational thinking practices in teacher development courses. In: E. K. Hawthorne, M. A. Pérez-Quiñones, S. Heckman, & J. Zhang (Eds.), *Proceedings of the 50th ACM technical symposium on computer science education* (pp. 976–982). ACM. https://doi.org/10.1145/3287324.3287470

Korkmaz, Ö., Çakir, R., & Özden, M. Y. (2017). A validity and reliability study of the computational thinking scales (CTS). *Computers in Human Behavior*, *72*, 558–569. https://doi.org/10.1016/j.chb.2017.01.005

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

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

Leonard, A. E., Daily, S. B., Jörg, S., & Babu, S. V. (2021). Coding moves: Design and research of teaching computational thinking through dance choreography and virtual interactions. *Journal of Research on Technology in Education*, *53*(2), 159–119. https://doi.org/10.1080/15391523.2020.1760754

Libeskind-Hadas, R., & Bush, E. (2013). A first course in computing with applications to biology. *Briefings in Bioinformatics*, *14*(5), 610–617. https://doi.org/10.1093/bib/bbt005

Liu, C.-C., Chen, W.-C., Lin, H.-M., & Huang, Y.-Y. (2017). A remix-oriented approach to promoting student engagement in a long-term participatory learning program. *Computers & Education*, *110*, 1–15. https://doi.org/10.1016/j.compedu.2017.03.002

Looi, C.-K., How, M.-L., Longkai, W., Seow, P., & Liu, L. (2018). Analysis of linkages between an unplugged activity and the development of computational thinking. *Computer Science Education*, *28*(3), 255–279. https://doi.org/10.1080/08993408.2018.1533297

Lu, J. J., & Fletcher, G. H. L. (2009). Thinking about computational thinking. *ACM SIGCSE Bulletin*, *41*(1), 260–264. https://doi.org/10.1145/1539024.1508959

Luo, F., Antonenko, P. D., & Davis, E. C. (2020). Exploring the evolution of two girls' conceptions and practices in computational thinking in science. *Computers & Education*, *146*, 103759. https://doi.org/10.1016/j.compedu.2019.103759

Lye, S. Y., & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in Human Behavior*, *41*, 51–61. https://doi.org/10.1016/j.chb.2014.09.012

Menekse, M. (2015). Computer science teacher professional development in the United States: A review of studies published between 2004 and 2014. *Computer Science Education*, *25*(4), 325–350. https://doi.org/10.1080/08993408.2015.1111645

Moher, D., Liberati, A., Tetzlaff, J., & Altman, D. G. (2009). Preferred reporting items for systematic reviews and meta-analyses: The PRISMA statement. *PLoS Medicine*, *6*(7), e1000097. https://doi.org/10.1371/journal.pmed.1000097

Moreno-León, J., Robles, G., & Román-González, M. (2015). Dr. Scratch: Automatic analysis of scratch projects to assess and foster computational thinking. *Revista de Educación a Distancia*, *46*(10), 1–23. https://doi.org/10.6018/red/46/10

Mouza, C., Pan, Y.-C., Yang, H., & Pollock, L. (2020). A multiyear investigation of student computational thinking concepts, practices, and perspectives in an after-school computing program. *Journal of Educational Computing Research*, *58*(5), 1029–1056. https://doi.org/10.1177/0735633120905605

Munoz, R., Villarroel, R., Barcelos, T. S., Riquelme, F., Quezada, A., & Bustos-Valenzuela, P. (2018). Developing computational thinking skills in adolescents with autism spectrum disorder through digital game programming. *IEEE Access*, *6*, 63880–63889. https://doi.org/10.1109/access.2018.2877417

Murphy, P. K., & Alexander, P. A. (2000). A motivated exploration of motivation terminology. *Contemporary Educational Psychology*, *25*(1), 3–53. https://doi.org/10.1006/ceps.1999.1019

Nam, K. W., Kim, H. J., & Lee, S. (2019). Connecting plans to action: The effects of a card-coded robotics curriculum and activities on Korean kindergartners. *The Asia-Pacific Education Researcher*, *28*(5), 387–397. https://doi.org/10.1007/s40299-019-00438-4

Nardelli, E. (2019). Do we really need computational thinking? *Communications of the ACM*, *62*(2), 32–35. https://doi.org/10.1145/3231587

National Research Council. (2013). *Next generation science standards: For states, by states*. The National Academies Press. https://doi.org/10.17226/18290

Newton, K. J., Leonard, J., Buss, A., Wright, C. G., & Barnes-Johnson, J. (2020). Informal STEM: Learning with robotics and game design in an urban context. *Journal of Research on Technology in Education*, *52*(2), 129–147. https://doi.org/10.1080/15391523.2020.1713263

Noh, J., & Lee, J. (2020). Effects of robotics programming on the computational thinking and creativity of elementary school students. *Educational Technology Research and Development*, *68*(1), 463–484. https://doi.org/10.1007/s11423-019-09708-w

Papavlasopoulou, S., Sharma, K., & Giannakos, M. N. (2020). Coding activities for children: Coupling eye-tracking with qualitative data to investigate gender differences. *Computers in Human Behavior*, *105*, 105939. https://doi.org/10.1016/j.chb.2019.03.003

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

Popat, S., & Starkey, L. (2019). Learning to code or coding to learn? A systematic review. *Computers & Education*, *128*, 365–376. https://doi.org/10.1016/j.compedu.2018.10.005

Price, C. B., & Price-Mohr, R. M. (2018). An evaluation of primary school children coding using a text-based language (java). *Computers in the Schools*, *35*(4), 284–301. https://doi.org/10.1080/07380569.2018.1531613

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

Repenning, A. (2012). Programming goes back to school. *Communications of the ACM*, *55*(5), 38–40. https://doi.org/10.1145/2160718.2160729

Repenning, A., Webb, D., & Ioannidou, A. (2010). Scalable game design and the development of a checklist for getting computational thinking into public schools. In: G. Lewandowski, S. Wolfman, T. J. Cortina, & E. L. Walker (Eds.), *Proceedings of the 41st ACM technical symposium on computer science education* (pp. 265–269). ACM. https://doi.org/10.1145/1734263.1734357

Repenning, A., Webb, D. C., Koh, K. H., Nickerson, H., Miller, S. B., Brand, C., Horses, I. H. M., 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*(2), 1–31. https://doi.org/10.1145/2700517

Rodríguez del Rey, Y. A., Cawanga Cambinda, I. N., Deco, C., Bender, C., Avello-Martínez, R., & Villalba-Condori, K. O. (2021). Developing computational thinking with a module of solved problems. *Computer Applications in Engineering Education*, *29*(3), 506–511. https://doi.org/10.1002/cae.22214

Román-González, M., Pérez-González, J.-C., & Jiménez-Fernández, C. (2017). Which cognitive abilities underlie computational thinking? Criterion validity of the computational thinking test. *Computers in Human Behavior*, *72*, 678–691. https://doi.org/10.1016/j.chb.2016.08.047

Rothstein, H. R., & Hopewell, S. (2009). Grey literature. In: H. Cooper, L. V. Hedges, & J. C. Valentine (Eds.), *The handbook of research synthesis and meta-analysis* (2nd ed., pp. 103–125). Russell Sage Foundation.

Rowe, E., Asbell-Clarke, J., Baker, R., Gasca, S., Bardar, E., & Scruggs, R. (2018). Labeling implicit computational thinking in pizza pass gameplay. In: R. Mandryk, M. Hancock, M. Perry, & A. Cox (Eds.), *Extended abstracts of the 2018 CHI conference on human factors in computing systems* (pp. 1–6). ACM. https://doi.org/10.1145/3170427.3188541

Scherer, R., Siddiq, F., & Viveros, B. S. (2019). The cognitive benefits of learning computer programming: A meta-analysis of transfer effects. *Journal of Educational Psychology*, *111*(5), 764–792. https://doi.org/10.1037/edu0000314

Selby, C., & Woollard, J. (2013). Computational thinking: *T*he developing definition. https://eprints.soton.ac.uk/356481/

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

Singer, L. M., & Alexander, P. A. (2017). Reading on paper and digitally: What the past decades of empirical research reveal. *Review of Educational Research*, *87*(6), 1007–1041. https://doi.org/10.3102/0034654317722961

Sung, W., & Black, J. B. (2020). Factors to consider when designing effective learning: Infusing computational thinking in mathematics to support thinking-doing. *Journal of*

Research on Technology in Education. Advance online publication. https://doi.org/10.1080/15391523.2020.1784066

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

Taylor, K., & Baek, Y. (2019). Grouping matters in computational robotic activities. *Computers in Human Behavior*, *93*, 99–105. https://doi.org/10.1016/j.chb.2018.12.010

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

Tran, Y. (2019). Computational thinking equity in elementary classrooms: What third-grade students know and can do. *Journal of Educational Computing Research*, *57*(1), 3–31. https://doi.org/10.1177/0735633117743918

Uzumcu, O., & Bay, E. (2020). The effect of computational thinking skill program design developed according to interest driven creator theory on prospective teachers. *Education and Information Technologies*. Advance online publication. https://doi.org/10.1007/s10639-020-10268-3

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

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

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

Witherspoon, E. B., Schunn, C. D., Higashi, R. M., & Shoop, R. (2018). Attending to structural programming features predicts differences in learning and motivation. *Journal of Computer Assisted Learning*, *34*(2), 115–128. https://doi.org/10.1111/jcal.12219

Wong, G. K., & Jiang, S. (2018). Computational thinking education for children: Algorithmic thinking and debugging. In: M. J. W. Lee, S. Nikolic, M. Ros, J. Shen, L. C. U. Lei, G. K. W. Wong, & N. Venkatarayalu (Eds.), *2018 IEEE international conference on teaching, assessment, and learning for engineering* (pp. 328–334). IEEE. https://doi.org/10.1109/TALE.2018.8615232

Xia, L., & Zhong, B. (2018). A systematic review on teaching and learning robotics content knowledge in K-12. *Computers & Education*, *127*, 267–282. https://doi.org/10.1016/j.compedu.2018.09.007

Yadav, A., Hong, H., & Stephenson, C. (2016). Computational thinking for all: Pedagogical approaches to embedding 21st century problem solving in K-12 classrooms. *TechTrends*, *60*(6), 565–568. https://doi.org/10.1007/s11528-016-0087-7

Yadav, A., Krist, C., Good, J., & Caeli, E. N. (2018). Computational thinking in elementary classrooms: Measuring teacher understanding of computational ideas for teaching science. *Computer Science Education*, *28*(4), 371–400. https://doi.org/10.1080/08993408.2018.1560550

Yadav, A., Mayfield, C., Zhou, N., Hambrusch, S., & Korb, J. T. (2014). Computational thinking in elementary and secondary teacher education. *ACM Transactions on Computing Education*, *14*(1), 1–16. https://doi.org/10.1145/2576872

Yin, Y., Hadad, R., Tang, X., & Lin, Q. (2020). Improving and assessing computational thinking in maker activities: The integration with physics and engineering learning. *Journal of Science Education and Technology*, *29*(2), 189–126. https://doi.org/10.1007/s10956-019-09794-8

Yu, J., & Roque, R. (2019). A review of computational toys and kits for young children. *International Journal of Child-Computer Interaction*, *21*, 17–36. https://doi.org/10.1016/j.ijcci.2019.04.001

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

Zhong, B., Wang, Q., & Chen, J. (2016). The impact of social factors on pair programming in a primary school. *Computers in Human Behavior*, *64*, 423–431. https://doi.org/10.1016/j.chb.2016.07.017

Author Biographies

Ndudi O. Ezeamuzie is a doctoral student in the Faculty of Education, The University of Hong Kong. He is also a computer scientist with research interest in computational thinking and programming in the context of STEM education.

Jessica S. C. Leung is an assistant professor in the Faculty of Education, The University of Hong Kong. Her current research interest, among other things, is epistemic practice and its development, in particular, within the context of science and STEM education.