International Journal of STEM Education

REVIEW **Open Access**

Integration of computational thinking in K-12 mathematics education: a systematic review on CT-based mathematics instruction and student learning

Huiyan Ye[1], Biyao Liang[2], Oi-Lam Ng[1*] and Ching Sing Chai[1]

Abstract
There has been substantial research undertaken on the integration of computational thinking (CT) in K-12 mathematics education in recent years, particularly since 2018 when relevant systematic reviews were conducted on the topic. Many empirical studies in this area have yet to elaborate clearly and explicitly on how CT may support mathematics learning, or otherwise, in CT-based mathematics activities. Addressing this research gap, we conducted a systematic review on the integration of CT in K-12 mathematics education with a focus on CT-based mathematics instruction and students learning under such instruction. The Web of Science database was searched for in terms of studies published from 2006 to 2021, from which 24 articles were selected to provide illustrations of CT-based mathematics instruction and related student learning, and they were further analyzed according to education levels and contexts, programming tools, learning outcomes in CT and mathematics, and the mutual relationship between CT and mathematics learning. Among the results, this review found that geometrized programming and student-centered instructional approaches were facilitators of productive learning in CT and mathematics. Moreover, CT-based mathematics learning entails an interactive and cyclical process of reasoning mathematically and reasoning computationally, which can occur when: (1) applying mathematics to construct CT artefacts; (2) applying mathematics to anticipate and interpret CT outputs; and (3) generating new mathematical knowledge in parallel with the development of CT. The findings contribute to an in-depth understanding of what, and how, CT-based mathematics instruction impacts student learning in K-12 contexts.

Keywords Computational thinking, Mathematics education, K-12, Systematic review

Introduction

In the last two decades, considerable literature has grown around the theme of computational thinking (CT), namely, “the thought processes involved in formulating problems and their solutions, so that the solutions are represented in a form that can be effectively carried out by an information-processing agent” (Wing, 2011, p. 20). First to coin the term, Wing (2006, 2008) emphasized that CT is a fundamental skill required in daily life and a kind of analytical thinking that shares close connections with mathematical thinking, engineering and scientific practices. Research on CT has received increasing attention, especially regarding its integration into K-12 educational contexts (e.g., Feldhausen et al., 2018; Jocius et al., 2021; Sırakaya et al., 2020; Swaid, 2015). One kind of such integration that has obvious connection, yet little is known

*Correspondence:
Oi-Lam Ng
oilamn@cuhk.edu.hk
[1] Department of Curriculum and Instruction, Faculty of Education, The Chinese University of Hong Kong, Shatin, N.T., Hong Kong
[2] Faculty of Education, The University of Hong Kong, Pokfulam, Hong Kong

about, is CT's integration in mathematics instruction and its associated learning outcomes (e.g., Barr & Stephenson, 2011; Sneider et al., 2014; Weintrop et al., 2016).

Using CT and programming tools to learn mathematics can be traced back to Papert's *Logo* programming (1980) and further developed by Weintrop et al. (2016), who illustrate the resemblance of CT practices and those of science and mathematics practices, i.e., data handling, modelling and simulation, computational problem-solving, and systems thinking. With CT-based mathematics instructional approaches becoming more widely adopted in K-12 contexts in recent years (e.g., Miller, 2019; Pei et al., 2018; Shumway et al., 2021), a number of design-based research have been conducted to improve the conceptual basis and practice of a "computationally enhanced mathematics education" (Ng & Cui, 2021, p. 848). Related studies have also advanced the instructional design, including the programming tasks to facilitate students' mathematics learning. These studies encompass Kotsopoulos et al.'s (2017) pedagogical framework for CT activities, designing for unplugged, tinkering, making, remixing in K-12 mathematical problem-solving in the domains of combinatorics (e.g., De Chenne & Lockwood, 2022), algebra (Bråting & Kilhamn, 2021), number theory and mathematical modelling (e.g., Ng & Cui, 2021) and geometry (e.g., Miller, 2019). Significantly, these studies evidence the reciprocal relationship between computer science and mathematical concepts, in the sense that the application of CT deepens mathematical disciplinary knowledge, while the context of CT-based mathematics learning improves one's CT skills (Pei et al., 2018). However, as argued by Lockwood and De Chenne (2020), while programming seems to be effective in learning certain mathematical topics, it cannot be concluded that it would be superior to paper-and-pencil, nor that all domains of mathematics are generally suitable for integration with computing. In addition, students were found to experience various challenges due to potential differences between the two types of thinking (i.e., computational and mathematical) when solving mathematical problems in programming contexts (Cui & Ng, 2021). Thus, there remains a lack of consensus in the interrelationship and components of overlap between CT and mathematics, especially surrounding the K-12 education contexts in which strong emphasis has been placed on CT development in age-appropriate and interdisciplinary ways (Hong Kong Curriculum Development Council, 2020; International Society for Technology in Education, 2016; National Research Council, 2013). An important question to be further explored is to clarify how computing can be effectively used to foster K-12 mathematics thinking and learning (Hickmott et al., 2018), and vice versa.

While existing reviews have explored the multi-faceted linkage of CT and K-12 education (Barcelos et al., 2018; Grover & Pea, 2013; Hickmott et al., 2018; Hsu et al., 2018; Tang et al., 2020), a scarcity of reviews has discussed the linkage or interplay between CT and K-12 mathematics thinking and learning (Shumway et al., 2021). The most recent and relevant review was conducted by Hickmott et al. (2018), which was a scoping review of how CT has been applied in K-12 mathematics classrooms broadly. The categories of analyses were mainly descriptive, including implicit or explicit connections to domains of mathematics, activity approaches, and evidence of impact on student learning (by quantitative methods of studies). Hickmott et al. (2018) highlighted the need for "studies that explicitly linked the learning of mathematics concepts with computational thinking were uncommon in the reviewed literature" (p. 65). In response, this paper aims to provide an in-depth examination of extant studies of K-12 practices that explicitly link CT with mathematics thinking and learning. In this way, we contribute toward understanding and synthesizing the interplay between CT and mathematics learning as evident in empirical studies, especially given the vast number of studies that have been published since Hickmott et al.'s (2018) review.

In this paper, we report on a systematic review with the dual purpose of reviewing broadly the characteristics of CT-based mathematics instruction and student learning under such environment, and then provide an in-depth analysis and discussion of the interplay between CT and mathematics learning outcomes from extant empirical studies. To do so, we conducted a review of studies that detail both teaching and learning processes in CT-based mathematics activities. It addresses the research gap regarding the impact of CT-based mathematics instruction on student learning, and also more clearly and explicitly identify the interplay between CT and mathematics learning. To achieve our dual foci, we pose the following four research questions (RQs):

1. In what educational contexts is CT integrated in K-12 mathematics education?
2. What tools and instructional approaches are used in CT-based mathematics activities?
3. What theoretical constructs or conceptual frameworks have been used, and what students' learning outcomes are characterized in CT-based mathematics activities?
4. What kinds of interplay between CT and mathematics thinking and learning are observed?

Methodology

We used the PRISMA systematic review method (Page et al., 2021a, b) to review the impact of CT-based mathematics instruction on student learning. In what follows, we describe the search process, selection process and method of data analysis.

Search process

The Web of Science (WOS) database was used as the source for literature search. We used "computational thinking" AND "math*" AND ("school*" OR "educat*") AND ("primary" OR "elementary" OR "secondary" OR "middle" OR "high" OR "K-12") as the search terms to search in "topic", which included searching in titles, abstracts, author keywords, and keywords plus. In consistent with other systematic reviews on CT (Barcelos et al., 2018; Hsu et al., 2018), we limited our search to the publication date starting from 2006, when Jeannette Wing first introduced the term CT; as such, the date range of the publications searched was between 2006-01-01 and the date we performed the search, i.e., 2021-11-15. With WOS automatically removing duplicate articles, we obtained 290 articles in total for further screening.

Selection process

The overall goal of the selection process was to improve relevance of the searched articles as pertaining to the goals of the current review. Our selection process consisted of three stages, and the procedures are summarized in Fig. 1. In the following, we discuss each stage in detail.

At Stage 1, non-English articles and non-journal articles (e.g., Conference Papers/Abstract/Unpublished papers) were removed to ensure language accessibility and quality of research of the literature pool, resulting in 164 remaining papers. We excluded unpublished articles because of the need for peer review to ensure quality. Consistent with similar review (Tang et al., 2020), our decision to exclude conference papers and abstract was based on the following reasons: (a) according to Bradford's Law (Testa, 2009), the majority of noteworthy scientific findings have been disclosed in journal articles. Conference papers in this situation would present findings that were similar to those in journal publications. (b) Our systematic review aimed to examine detailed teaching process or student learning process. In general, conference papers did not report such information due to the limited space.

In the next two stages, we developed screening criteria as aligned with our research purpose to investigate the integration of CT in K-12 mathematics education. Although we included keywords of "K-12", "CT", and "math" in the search terms, there was no guarantee that the search records were relevant to those terms, as we only limited the search in titles, abstracts, and keywords. Therefore, at Stage 2, we further reviewed the title, abstract, and full text (when necessary) holistically to eliminate studies that were not conducted in K-12 contexts (e.g., at collegiate levels) or were not focused on CT and/or mathematics education. For example, some articles only mentioned the term "STEM (Science, Technology, Engineer and Mathematics)", "mathematics", or "CT" in the abstracts, but the primary focus of the studies was neither CT-related nor mathematics education-related, i.e., these articles are not about the teaching, learning, and affect issues of CT and mathematics. After the screening of this stage, we removed 111 articles, and the remaining 53 articles entered the full-text screening at the next stage.

At Stage 3, we aimed to screen for those studies that provide adequate descriptions of the teaching and learning processes to shed light on the relationship between CT-based mathematics instruction and student learning outcomes in CT and mathematics. First, the articles must report on empirical studies, so that the quality of student learning with reference to the instructional design can be empirically investigated. As such, we excluded theoretical papers, reviews, and articles that described provisional activities without implementation. Second, we identified studies that analyzed student learning processes and outcomes upon engaging in some form of CT-based mathematics instruction. This means we excluded studies that did not offer insight into features of student learning, such as research focusing only on teachers' learning (e.g., Bouck et al., 2021), and articles that only described the process of implementation or design of a course, task, or learning program, without explicitly reporting on student learning (e.g., Matsumoto & Cao, 2017). Regarding the third and final criterion, we continued to determine whether the article explicitly described any features of the CT-based mathematics instructional process (e.g., tools, tasks, instructional approaches); this inclusion criterion directly relates to our research questions. The first author and the second author independently conducted screening against these criteria. For discrepant screening results, the authors discussed each one until mutual agreement. This screening stage was concluded with 24 articles that constituted the final collection of review articles.

Data analysis

Although our research questions seem to be more inclined to retain the studies using qualitative research methods, the numerical results (Fig. 2, left) does not show significant difference in the use of the three research methods (qualitative at 42%, quantitative at

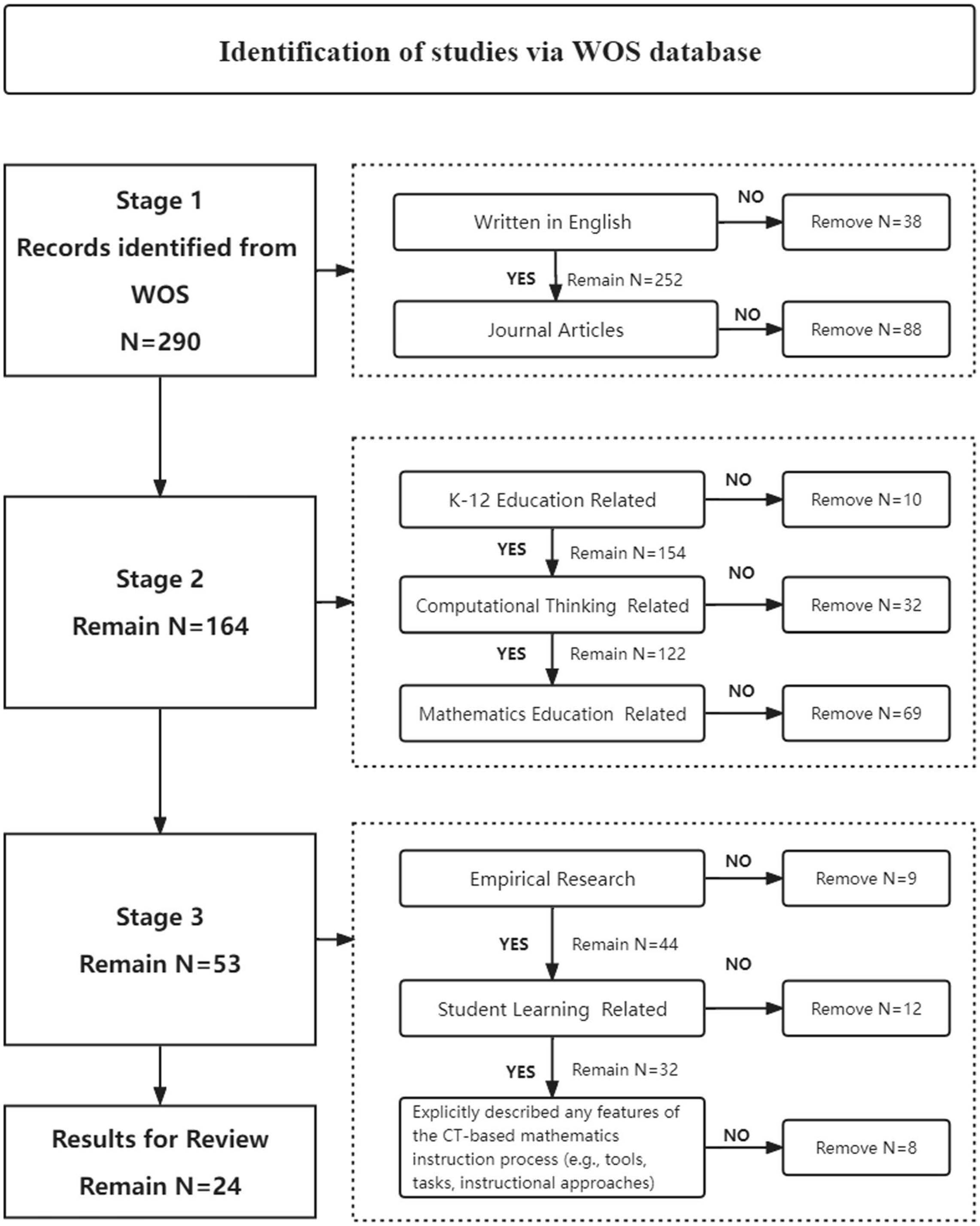


Fig. 1 Paper selection process

33%, mix method at 25%). We further categorized the data collection methods (Fig. 2, right) and found that observation and test are the two most frequent ways for assessing and having access to student learning.

Upon completion of the screening process, the first and the third author initially proposed a set of codes which addressed the RQs upon their reading of the 24 articles. For example, in response to answer RQ1 (educational

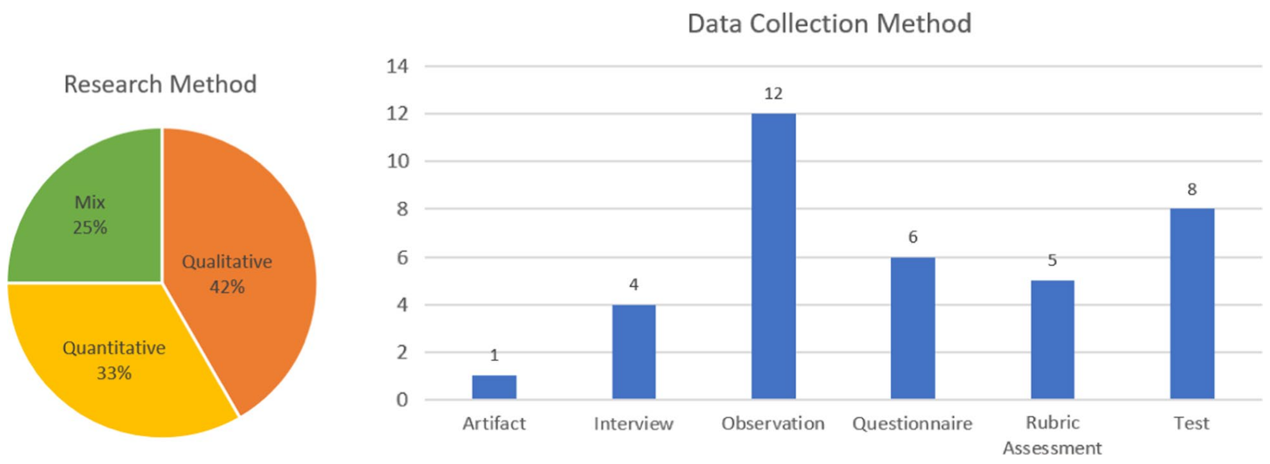


Fig. 2 Research methods and data collection methods undertaken by the reviewed studies

context), the deductive codes included education level (grade), educational setting (school education or after-school activity), and disciplinary context (multidisciplinary or CT-based mathematics context). Likewise, in response to RQ2 and RQ3, the authors prescribed codes that addressed the programming tools and instructional approach adopted, as well as learning outcomes (associated with CT or mathematics).

To arrive at the second- and third-level codes, the first and third author performed constant comparative analysis (Strauss & Corbin, 1990) to identify and refine the codes that adequately describe the methods and results highlighted in the empirical studies until saturation of categories. Specifically, the authors independently coded six of the articles (or 25% of the selected articles) and discussed the coding results to (1) operationalize the meaning of each code; and (2) delete, add, and synthesize coding items that were helpful to answer the RQs. This process yielded the second- and third-level codes, and a finalized coding scheme with code definitions as presented in Table 1. In the case of RQ4, the authors used constant comparative analysis (Strauss & Corbin, 1990) to obtain new findings regarding the interplay between CT and mathematics. Open coding was first used to create initial codes by extracting information from the literature that was pertinent to student learning. Then, the authors practiced axial coding and selective coding to create a core category to connect them and achieved the final codes.

Upon completion of the coding scheme, the authors coded another eight articles independently, and the inter-rater agreement exceeded 90%. At last, the first author coded all the remaining articles, and when encountering uncertain coding, all authors discussed until agreement. We report on the coding results in the next section.

Result

In this section, we first describe the educational context, where CT-based mathematics teaching and learning activities are implemented (section "Educational context of CT-based mathematics instruction"); this includes reporting on the education level, educational setting, and disciplinary context of the reviewed studies. Then, we discuss the pedagogical aspects of the articles (section "Pedagogical aspects of CT-based mathematics instruction"), which include the tools and instructional approaches used during CT-based mathematics instruction. Finally, we review students' learning outcomes in relation to CT and mathematics (section "Interplay between CT and mathematics learning outcomes"), both in terms of the diverse constructs used to characterize student learning, as well as the interplay between CT and mathematics learning outcomes as evident in the reviewed literature. We discuss the results along with our further insights in each sub-section.

Educational context of CT-based mathematics instruction

To begin, we use Fig. 3 to summarize the literature count regarding the education level in which the studies are conducted. Notably, more researchers have carried out studies in primary educational contexts, followed by lower secondary education. This may be related to researchers' attention to the importance of students' early exposure to CT (Jurado et al., 2020; Sáez-López et al., 2019) alongside the emergence of child-friendly programming tools for easy CT experimentation during a critical cognitive developmental period for CT learning (Rodríguez-Martínez et al., 2020). Among the 24 reviewed articles, five included research participants across education levels; these studies highlight the

Table 1 Coding scheme

First-level code	Second-level code	Third-level code	Code definition
Educational level	Early childhood	n/a	Kindergarten
	Primary	n/a	Grades 1–6
	Secondary	Lower-secondary	Grades 7–9
		Upper-secondary	Grades 10–12
Educational setting	Formal	n/a	Compulsory; intentional and structured
	Non-formal	n/a	Voluntary; intentional and structured
Disciplinary context	Multidisciplinary	n/a	Learning activities integrate several disciplines and/or focus on solving CT-multidisciplinary or interdisciplinary problems, e.g., STE(A)M; math and science
	Mathematical (disciplinary) contexts	n/a	Learning activities integrate only mathematical topics and/or focus on solving problem in mathematics only
Learning tool	Screen-based programming	Text-based programming	Require strict, precise, and complex syntactic rules to code and often requiring a keyboard to input text
		Block-based programming	Can drag and drop the visual script blocks on the screen with a mouse or their hands to program
		Geometrized Programming	Programming elements are geometric objects, the grammar and rules are the properties of geometric objects. Only when the properties of geometric objects are satisfied can the correct structure of geometric figure be constructed
	Tangible programming	n/a	Requires coders to use hardware objects with physical user interface to build programs

Table 1 (continued)

First-level code	Second-level code	Third-level code	Code definition
Instructional approach	Task structure	Problem-based learning	Learner takes a problem as the starting point, focuses on the inquiry process and finally proposes solutions to the problem (Savery, 2015)
		Project-based learning	The goal of project-based learning is to produce "products" with specifications. It is a complex project, where students encounter multiple problems and need to make clear plans and reasonable decisions to satisfy the requirements of the project (Blumenfeld et al., 1991)
		Inquiry-based learning	Learning begins with questions and is supported by a series of mathematical tasks. Teachers can scaffold students to solve problems by providing related information. (Ernst et al., 2017)
	Other structure	Embodied learning	Teachers design concrete activities for students to participate with hand or body movement. Students are expected to experience abstract concepts with gestural or bodily experiences
		Game-based learning	Learning during gameplay (embodied games or digital games)
		Pair- or group-learning	Two or more students in different roles (e.g., one is "driver", who is responsible for operating corresponding tools; the other is an "observer," who observe and give comments and suggestions) form a group to collaborate on learning
Learning outcome	CT concepts	For details, see section "Computational concepts and practices associated with CT-based mathematical activities"	
	CT practices		
	Mathematical topics	For details, see section "Mathematical learning outcomes demonstrated in computational contexts"	
Interplay	For details, see section. "Mathematical learning outcomes demonstrated in computational Contexts"		

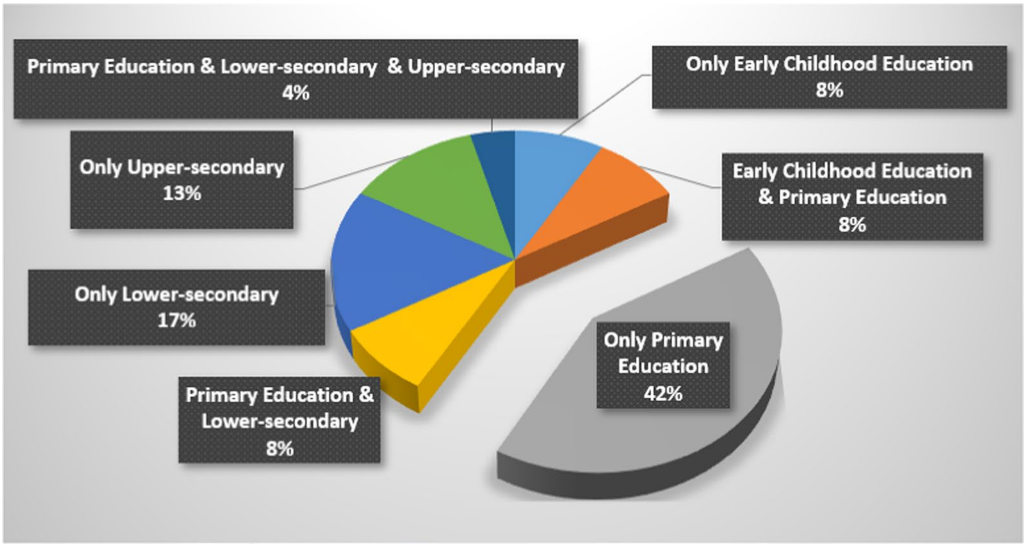


Fig. 3 Literature count of each education level

feasibility of conducting CT-based mathematics activities to students with varying knowledge bases and skill levels. The cross-grade activities used among these studies are afforded by the spiral mathematics curricula across K-12 education, such as the recurrent arithmetic topics in kindergarten and lower primary education (Sung et al., 2017), the relevant transition from arithmetic to algebraic thinking in the middle school grades (Ng & Cui, 2021), and the interrelated 2D geometry topics across secondary education (Wilkerson-Jerde, 2014). In addition, these studies commonly provided an open-ended constructionist learning environment which involved the students creating CT artefacts with little direct instruction (Papert & Harel, 1991), during which the students could explore and develop non-prescribed learning outcomes in both mathematics and CT. As argued by Muñoz et al. (2020) and Ng et al. (2021), CT-based mathematics activities are suitable for cross-grade teaching and learning, because they support non-traditional curricula learning, with the former study showing that mathematics learning could take on a range of domains with little prior mathematics knowledge necessary, and the latter study concluding that the developmental sequence of learning mathematics maybe altered by the use of computational tools. In particular, Ng et al. (2021) highlights the conceptual connection between skip counting and geometric sequences, which takes roughly 7 years to learn in traditional curricula; but in the Scratch programming environment, it is as straightforward as replacing the symbol for operation. In summary, we synthesize from the reviewed studies the potential for CT-based mathematics instruction to achieve cross-grades mathematics learning. At the same time, they also stimulate questions regarding the constraints of disciplinary mathematics learning within a highly structured curricula which is predominantly intended for using paper-and-pencil as the medium of learning. As such, we also draw attention to the constraints of research and practice in CT-based mathematics instruction given the current curricula structure.

As part of educational context, we analyzed the educational settings to understand whether CT-based mathematics instruction has been applied in formal or informal education, with the former being compulsory and the latter being voluntary. We found that 17 reviewed studies were conducted within a formal schooling environment, and 7 studies were conducted outside of formal education settings (of which 4 were in primary education and 3 were in secondary education). In particular, the 7 studies were contextualized in after-school programs or summer camps, which are also considered as *non-formal education*, since they are intentional and structured when compared to spontaneous and unstructured activities in daily informal settings (Eshach, 2007). We also investigated the disciplinary context of CT-based mathematics instructions to examine whether a single or multiple disciplines were involved in the learning activities. We found that a minority of studies ($n=7$) aimed to develop or support students' CT and mathematics learning in an interdisciplinary or multidisciplinary way (e.g., STEM, STEAM, Math + Science). For example, the study of Gilchrist et al. (2021) was situated in a pandemic awareness STEM outreach curriculum focused on medicine, mathematics and technology, where students developed an epidemic outbreak simulator using the Python programming

Table 2 Tools used in CT-based mathematics instruction

Types of Tools		Example
Screen-based Programming [18]	Text-based programming [6]	*CoCalc* with *Python*; *Processing;* spreadsheet [2]; *MATLAB; MaLT2*
	Block-based programming [9]	*Arduino* [3]; *ViMAP*; *Scratch* [3]; iPad-based *Scratch Jr*; *mBot* with *mBlock*; *Robot Dash*
	Geometrized Programming [3]	*Sketchpad*; *Lattice Land*; *Moodle-G platform* with *GeoGebra* functions integrated
Tangible Programming [3]		*KIBO robot 18 Kit; Bee-Bot robot; Botley; Cubetto*

Note: [N] denotes total literature counts of each tool

with *CoCalc*. In another study, students were asked to use *KIBO* robots to solve STEAM-related problems in designing and engineering a real-life artefact (Jurado et al., 2020).

The remaining 17 articles reported on students' CT and mathematics learning in the context of solving CT-based mathematical problems (see more discussion in section "Interplay between CT and mathematics learning outcomes"). For example, Wilerson-Jerde (2014) explored how secondary students create and analyze fractal structure with *Categorizer*, while Psycharis and Kallia (2017) posed a statistical problem for students to solve with *MATLAB* programs. At the primary education level, students used *Scratch* to learn geometric figures by drawing squares and spirolaterals (Miller, 2019). It is evident that most practices of CT-based mathematics instruction are situated in mathematical (disciplinary) problem-solving contexts rather than in STEM (multidisciplinary) contexts. The lack of studies which address multidisciplinary learning in CT-based mathematics instructions is likely associated with our earlier findings that most studies were conducted in formal education context. Moreover, it has been well-documented in literature that STEM integration is difficult to practice, especially when the focus of STEM integration is on achieving mathematics learning (English, 2016). Together, the findings surrounding educational levels and settings prove that CT-related research in both STEM and mathematics education is at a beginning point, and that more empirical research is needed to inform effective integration of CT in formal K-12 curricula context.

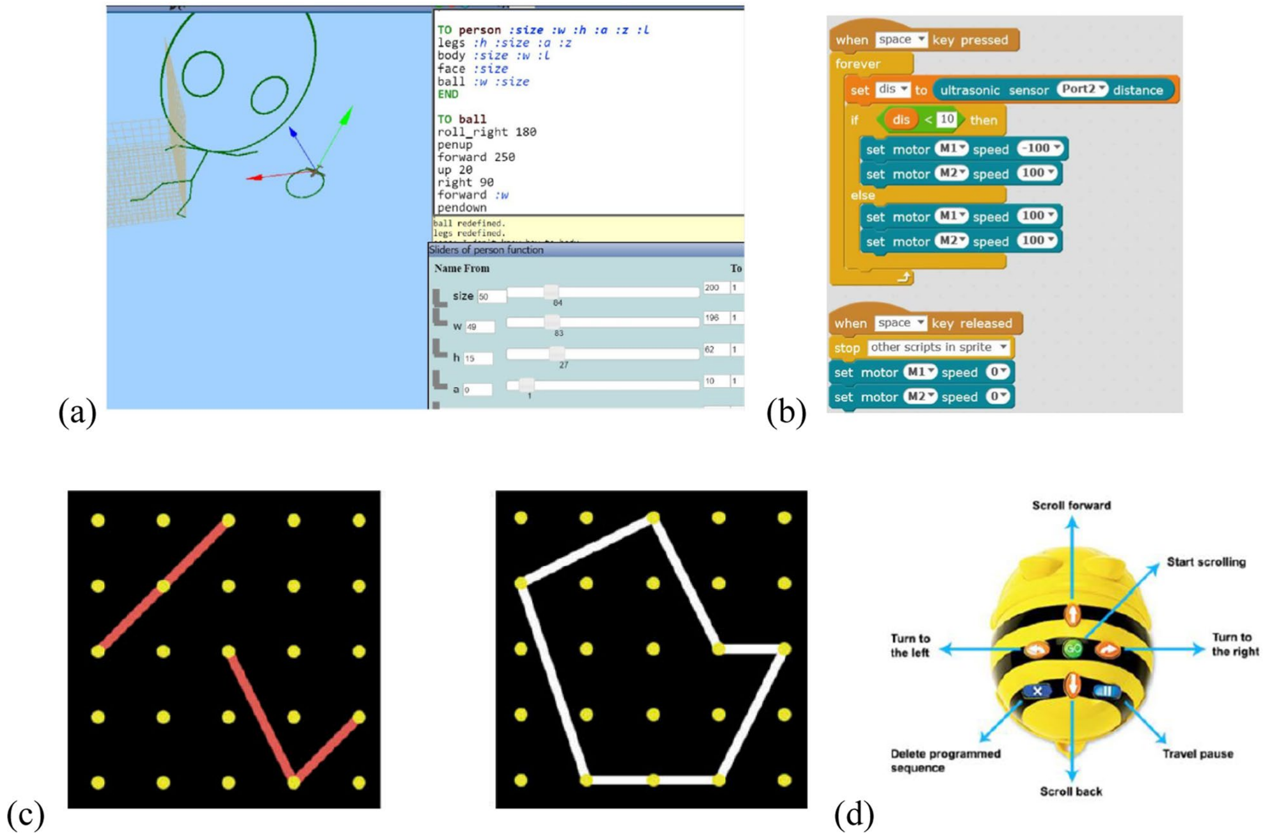
Pedagogical aspects of CT-based mathematics instruction

In this section, we focus on the pedagogical tools and approaches that have been developed and utilized to support CT-based mathematics instruction and student learning.

Tools

While CT is generally related to computer science and programming tools, CT learning activities are not limited to plugged activities (implemented with the use of computers). Our review shows that unplugged CT activities (implemented without the use of computer) are often carried out by researchers as precursory activities for plugged CT activities, such that students would experience some target CT concepts and practices in advance, and this is conducive to students' adaptation to programming (plugged) activities (Bouck & Yadav, 2020; Sung et al., 2017). In Bouck and Yadav's (2020) study, for example, an unplugged game of moving plastic cups on a surface was used to help students gain a better appreciation for what it is algorithms and debugging before students began coding to move a robot. In terms of plugged activities, we identified 18 different tools and platforms used in CT-based mathematics activities across the literature, with some being conventional tools for programming and technology instruction (e.g., *Python, Scratch, Robot Dash*), and some being more commonly used in mathematical practices (e.g., *Sketchpad, GeoGebra, MATLAB*). Based on the programming platform interface, we divide these tools into two categories: *screen-based programming* and *tangible programming* (Table 2).

Under screen-based programming, by which users program within a screen (e.g., computer screen), we identified three sub-categories of programming languages: *text-based programming, block-based programming,* and *geometrized programming*. We define text-based programming tools as those requiring strict, precise, and complex syntactic rules to code, thereby often requiring a keyboard to input text (Horn & Jacob, 2006), such as *Processing, Python, Matlab,* and *MaLT2* (Fig. 4a). In block-based programming environment (e.g., *Scratch, ViMAP*), students can drag and drop the visual script blocks on the screen with a mouse or their hands to program. These script blocks constitute more natural and intuitive ways to operate a program while providing useful visual cues on how and where to use program commands (Fig. 4b). Unique to CT-based mathematics instruction is a *geometrized programming language,* which, as Sinclair and Patterson (2018) proposed, is a third screen-based programming language whose elements are geometric objects rather than texts or blocks. One typical example is the Geometer's Sketchpad, and another example is *Lattice Land* (see Pei et al., 2018), whose interface enables users to interact directly with dots to construct segments



```
TO person :size :w :h :a :z :l
legs :h :size :a :z
body :size :w :l
face :size
ball :w :size
END

TO ball
roll_right 180
penup
forward 250
up 20
right 90
forward :w
pendown
```

```
ball redefined.
legs redefined.
```

Sliders of person function

Name	From		To	
size	50	84	200	1
w	49	83	196	1
h	15	27	62	1
a	0	1	10	1

```
when space key pressed
forever
    set dis to ultrasonic sensor Port2 distance
    if dis < 10 then
        set motor M1 speed -100
        set motor M2 speed 100
    else
        set motor M1 speed 100
        set motor M2 speed 100

when space key released
stop other scripts in sprite
set motor M1 speed 0
set motor M2 speed 0
```

Fig. 4 **a** Interface of text-based programming tool *MaLT2* (Grizioti & Kynigos, 2021, p. 10); **b** a block-based programming tool *mBot* (Sáez-López et al., 2019, p. 1414); **c** segments and a polygon drawn in a geometrized programming tool *Lattice Land* (Pei et al., 2018, p. 78); **d** a *Bee-bot robot*, a form of tangible programming used in Muñoz et al., (2020, p. 7)

and polygons and investigate lattice geometry as a form of geometrized programming (Fig. 4c).

Tangible programming requires coders to use hardware objects with physical user interface to build programs, as opposed to "write" or "drag-and-drop" on the computer screen. With tangible programming, young coders can create procedural syntax to perform physical actions such as moving between two locations by scanning a code sequence (*KIBO Robot*), pressing command buttons on the machine (*Bee-bot Robot*; Fig. 4d), entering commands with an external remote (*Botley*), and placing the desired tiles on the programming board (*Cubetto*).

Our review provides insights into the features of the programming tools and their implications for student learning, not only in terms of CT development in age-appropriate ways but also the quality of connections with learning content in mathematics. Often, the choice of programming tool in relation to students' characteristics, such as developmental stage, prior knowledge, and experience with the programming tool, were considerations taken by researchers in designing CT-based mathematics activities. For example, the study by Sáez-López et al. (2019) acknowledged the use of block-based programming as attempts to instigate CT without tedious coding. As such, they support kindergarten and primary students to exercise CT through activities with embedded computational concepts, such as variable, sequence, iteration (or loops), conditional statements and event handling (Miller, 2019; Ng & Cui, 2021; Sáez-López et al, 2019). These studies also caution that if the CT concepts and practices involved are not made explicit, it may be difficult to tell if students could transfer their knowledge gained to other problem-solving contexts. Text-based and geometrized programming demand high requirements on the programmer, such as understanding the programming syntax and having adequate mathematical knowledge, which were used frequently in secondary education. In turn, it also demands the instructor to be knowledgeable in these aspects when integrating CT into mathematics instructions, evident in mathematics teachers often needing to collaborate with computer programming teacher on CT-based mathematics instruction (Psycharis & Kallia, 2017).

Although extant studies have considered the cognitive demand required on the programming tool in age-appropriate ways during CT-based mathematics activities, few researchers have explicitly communicated the affordances of the programming tool in supporting the target mathematics learning content. In response, we aim to generate further understanding from the reviewed empirical studies on this question. First, the reviewed studies suggest that the physicality of tangible programming is supportive to the learning of number, measurement, geometric shapes and spatial concepts and skills (Jurado et al., 2020; Muñoz et al., 2020; Shumway et al., 2021). Sáez-López et al. (2019) further recommended the use of block-based programming and robots for learning number concepts in CT-based mathematics activities, saying "there are many advantages to teaching computational concepts, coordinates, values and integer numbers as motivation for the student to learn how to operate the robot. The immediate feedback and response of the robot when operating with numbers is a powerful and highly motivational tool for students" (p. 1422). Conversely, text-based programming requires higher level of abstraction; these features make it suitable to deal with more advanced number topics, such as statistics (Gilchrist et al., 2021; Psycharis & Kallia, 2017), number sequence (Chan et al., 2021), and algebraic relationships (Kaufmann & Stenseth, 2021). We note that although the spreadsheet (e.g., Excel) is not generally defined as a CT tool, it executes some statistical functions and output upon inputting text-based formulae, and thus, it was considered as a text-based CT tool in both studies of Chan et al. (2021) and Valovičová et al. (2020).

Uniquely in a geometrized programming environment, the grammar and rules for execution are the properties of geometric objects; only when the properties of geometric objects are satisfied can the correct structure of geometric figure be constructed. Studies which adopted geometrized programming would take a "point" (in a Euclidean sense) as the most basic programming element or input, from which one can generate a more complex output, such as constructing a segment by joining two points or further formed a quadrilateral by constructing four segments. As Sinclair and Patterson (2018) described, applying actions on geometric objects (e.g., dragging a point) is like executing a program, and the actual behavior of the objects provides feedback that helps the coder to debug or proceed to program. In summary, geometrized programming tool as a form of computing tool is especially congruent to the learning of geometry. If future development in CT tools can be specialized to be strongly associated with a particular form or learning content at the tool level, it will open a new venue of educators to support mathematics learning in CT contexts.

Instructional approaches

Regarding CT-based mathematics instruction, we identified several common instructional approaches, and we further sub-categorized them by task structure (i.e., the organization of learning tasks and learning processes): *problem-based learning* (Gilchrist et al., 2021; Kaufmann & Stenseth, 2021; Ng & Cui, 2021), *inquiry-based learning* (Gilchrist et al., 2021; Pei et al., 2018), and *project-based learning* (Sinclair & Patterson, 2018). While reviewing the studies, we also note other learning modes, such as *embodied learning* (Sung et al., 2017), *game-based learning* (Grizioti & Kynigos, 2021), and *pair-* or *group-learning* (Echeverría et al., 2019; Grizioti & Kynigos, 2021; Kaufmann & Stenseth, 2021; Muñoz et al., 2020). In the present review, we focus on analyzing the characteristics of problem-based learning, project-based learning, and inquiry-based learning, since they are more dominant in the literature.

In general, the studies adopting one of these task structures share the common characteristic of implementing student-centered approaches, thereby positioning students as active mathematics learners, computational thinkers and problem solvers. They are also constructivist- or constructionist-oriented, revealing the epistemological underpinning the movement of CT for mathematics. While sharing in common the goal of constructionist or "learning-by-making" (Papert & Harel, 1991, p. 1), we note that the kinds of "making" vary among the three task structures: "making" stands for creating a programmable solution in problem-based learning, constructing a CT artefact in project-based learning, and formulating complex explanations through explorations in inquiry-based learning. In the following, we first provide an example of how each task structure manifest under CT-based mathematics activities to illuminate patterns within and across them, followed by discussing their implications for advancing the integration of CT and mathematics in K-12 contexts.

To begin, our review points to the high compatibility of *problem-based learning* in CT-based mathematics activities, given the central connection that CT is a mode of problem solving and a kind of analytical thinking in common with mathematical problem solving and reasoning (Baldwin et al., 2013; Wing, 2008). In this context, the main objective is to apply CT and mathematical knowledge and skills—as they are developed—during the course of problem solving. An example is provided in Kaufmann and Stenseth (2021), who utilized problem-based learning by first introducing the computational tool, then providing a default program as the starting

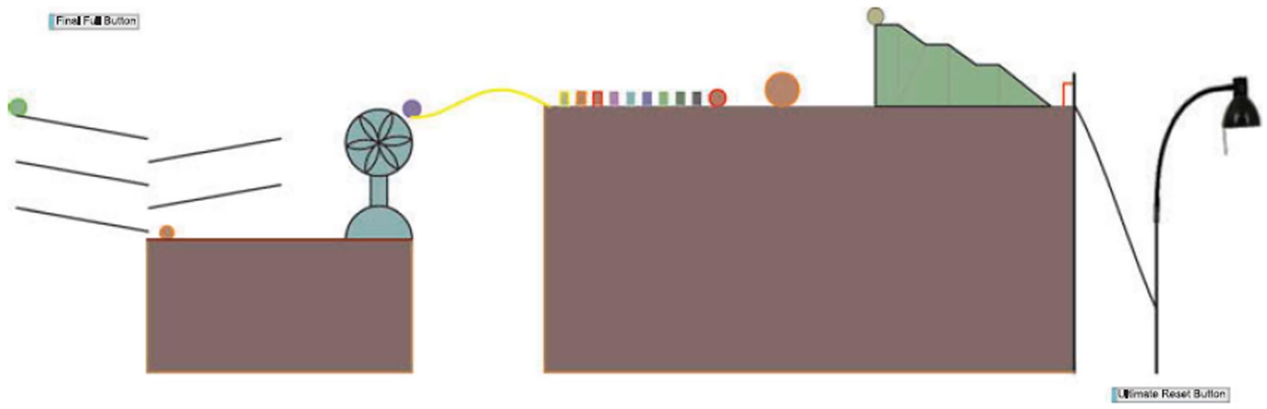


Fig. 5 Fan and Lightbulb Rube Goldberg machine (Sinclair & Patterson, 2018, p. 61)

problem for the students. Without providing any other scaffoldings, students grouped in pairs were asked to propose solutions to rectify the coding, so that the wheel can rotate at a constant speed and the size of the wheel can vary without affecting how the wheel rotates. This approach facilitates "autonomy and active learning experiences, to promote uncertainty and active methodologies centred on the student" (Sáez-López et al., 2019, p. 1406).

Different from problem-based learning, the goal of *project-based learning* is to produce CT artefacts with some project specifications; typically, the students encounter multiple problems related to CT or mathematics and need to make clear plans and reasonable decisions within their knowledge to satisfy the requirements of the project. Sinclair and Patterson (2018) illustrated a study adopting project-based learning, where a teacher first shared some examples of machines created with *Sketchpad*, led the class to brainstorm similar machines they could make, and provided instructions for the "Machine Project" with specific features (e.g., it should have animated features and involve trigonometry in some way). After students made a project proposal, they worked on their projects independently over the next 2 months and finally created a working machine with *Sketchpad* (see an example in Fig. 5).

Akin to the other two task structures, *inquiry-based learning* unfolds with an open question with no obvious solutions in CT-based mathematics context. Uniquely, this form of learning often led to new insights or discovery about mathematics by students upon being guided by a series of exploration and teacher scaffolding (Savery, 2015). This approach was adopted by Pei et al. (2018), who designed an activity in *Lattice Land* and asked students to explore the area of triangles. The teacher scaffolded students' inquiry-based learning by generating different shapes of triangles, so that students can move the vertices of triangles to identify relationships between triangular area and perimeter.

Of interests to our review is also to understand how specific task structure (problem-based, project-based, and inquiry-based) and programming tool (text-based, block-based, geometrized, and tangible) combine to offer opportunities for productive CT-based mathematics learning. We provide evidence to this question in several ways. First, problem-based learning is well-suited with screen-based programming to facilitate CT-based mathematics activities. This context situates the students as designing computer programs that execute the desirable solution to a problem which is usually open-ended, and, therefore, complementary to the open-ended characteristics of programming practices (Ng et al., 2021; Rodríguez-Martínez et al., 2020). Even if the desirable result is fixed, such as a numerical value, the programming environment often allows the student to take on different programming pathways and computational means to achieve the desirable result (Ng et al., 2021), as aligned with Papert's notion of computing tools as "flexible material" and "objects-to-think-with". Second, the affordances of non-text and multimodal programmable elements in some screen-based programming tools have proven to be powerful for students to present their artefacts during project-based learning. As in the case of mBlock, the students in Sáez-López et al.'s (2019) studies have the freedom utilize block-based programming to design what actions ("Motion", such as direction, speed) and sensory input ("Sensing", such as keyboard, obstacle detecting) to incorporate in the project. Third, geometrized programming enabled geometrical objects to be shown and visualized dynamically onscreen, thereby conveying a sense of temporality in students' thinking that is not available by paper-and-pencil (Sinclair & Patterson, 2018). Finally, another dimension of project-based learning is the aesthetics of expressing one's project, since the CT artefact

Table 3 Different frameworks of CT used in the reviewed articles

Framework	CT components	References
Weintrop et al. (2016)	Data practices, modeling and simulation practices, computational problem solving practices, and systems thinking practices	Cui and Ng (2021); Echeverría et al. (2019); Gilchrist et al. (2021); Ng and Cui (2021); Pei et al. (2018)
Rich et al. (2020)	Programming: sequence, repetition, conditionals	Cui and Ng (2021)
Brennan and Resnick (2012)	Computational concepts; computational practices; computational perspective	Ng and Cui (2021); Ng et al. (2021)
Bocconi et al. (2016)	(1) Abstraction (2) algorithmic thinking (3) automation (4) decomposition (5) debugging (6) generalization	Valovičová et al. (2020)
Román-González et al. (2017)	Computational concepts: sequences, iteration or loop, event handling, conditionals	Rodríguez-Martínez et al. (2020)
Hoyles and Noss (2015)	CT practices: abstraction, algorithmic thinking, decomposition, pattern recognition CT concepts: variable; conditions; subroutines; loops/iteration	Sinclair and Patterson (2018)
Berland and Wilensky (2015)	Computational perspectives	Sung et al. (2017)
Ho et al. (2021)	Decomposition, pattern recognition, abstraction, algorithmic design	Chan et al. (2021)

can be programmed to appear in different colours, sizes, and arrangements (for example, Scratch has its own paint editor and sound editor built-in to enhance the presentation of a program). These are important features of block-based and geometrized programming that support creative expressions in how the students' projects could be presented.

Interplay between CT and mathematics learning outcomes

When implementing CT-based mathematics activities, one assumption educators have been adopting is that the integration of computer science and mathematics education can generate affordances that a sole focus on one may not have. Naturally, a central question researchers have been inquiring is: how do students' CT and mathematics learning interact when they are involved in CT-based mathematics instruction, and what are the consequences of such interactions? While an intuitive and straightforward answer to this question is that the influences are bi-directional and beneficial for student learning, our analysis reveals non-trivial results—that there exist complex interplays and even mismatches when coordinating computational and mathematical modes of thinking. In the following sub-sections, we first review the theoretical constructs and conceptual frameworks used to characterize students' computational concepts and practices as associated with their CT-based mathematical activities. Then, we discuss the ways in which mathematics is used and learned in such contexts.

Computational concepts and practices associated with CT-based mathematical activities

Since the introduction of the notion of CT, researchers have put forward different views on the definition and components of CT (e.g., Brennan & Resnick, 2012; Hoyles & Noss, 2015; Weintrop et al., 2016). Among the 24 reviewed articles, 11 articles specified the CT definitions or frameworks used, as summarized in Table 3. Among these frameworks, Brennan and Resnick (2012) proposed one of the most influential framings of CT for K-12 education that consists of three dimensions: *computational concepts*, *computational practices*, and *computational perspectives*. We focus our following discussion on the first two dimensions as they are addressed more often in the literature.

Computational concepts We use Table 4 to summarize the CT concepts studied by prior researchers. In addition to providing the definitions commonly accepted in computer science and adopted in the reviewed literature, we offer our interpretations of how each computational concept is operationalized and illustrated in CT-based mathematics context. In doing so, we set out to create a theoretical tool for mathematics education researchers to integrate CT in their research and teaching, and more importantly, to continually enrich and refine the definitions in future research.

One prevalent CT concept mentioned in the literature is *variable.* In working with CT-based mathematics tasks, students often need to define a variable and establish its initial value, and then this variable should take on subsequent values when the program is run. Besides representing a quantity, the use of variables may also be extended to represent an algebraic expression (e.g., result of operating with two or more variables; Ng & Cui, 2021), or geometric location (e.g., locus of a point having a variable location; Sinclair & Patterson, 2018) which could be stored, retrieved and operated on. The observed range of

Table 4 Summary of computational concepts and how they are applied in CT-based mathematics context

CT concepts	CT concepts as defined in the reviewed literature	CT concepts as applied in CT-based mathematics context	Additional references from the reviewed literature
Variable	An entity that can "store, retrieve, and update values" (Ng & Cui, 2021, p. 6)	A numerical (e.g., number of times to carry out a specific mathematical procedure; Ng & Cui, 2021), algebraic (e.g., result of operating with two or more variables; Dickes et al., 2020; Kaufmann & Stenseth, 2021; Ng & Cui, 2021), or geometric (e.g., movement of a point having a variable location on a circle; Sinclair & Patterson, 2018) representation of an entity whose values can be stored, retrieved and operated on	None
Conditionals	"Instructions that either perform an action or not, according to a given condition" (Rodríguez-Martínez et al., 2020, p. 319)	A set of decisions, actions, statements executed based on numerical (e.g., when a number equals a certain value; Ng & Cui, 2021), algebraic (e.g., when a parametric equation determines the amount of rotation of the wheels; Ng & Cui, 2021; Sinclair & Patterson, 2018) or geometric properties (e.g., when a draggable point determines how much a given shape is scaled by; Sinclair & Patterson, 2018)	Jurado et al. (2020); Rodríguez-Martínez et al. (2020); Sáez-López et al. (2019)
Loops	"The control structure that makes it possible to repeat one or more sequences multiple times" (Rodríguez-Martínez et al., 2020, p. 319)	Repeatedly carry out a mathematical procedure, e.g., a numerical calculation (Cui & Ng, 2021), or geometric construction (Dickes et al., 2020; Jurado et al., 2020; Miller, 2019; Sinclair & Patterson, 2018)	Rodríguez-Martínez et al. (2020); Sáez-López et al. (2019)
Sequences	"The sequence or structure of instructions that should be followed to complete a goal" (Rodríguez-Martínez et al., 2020, p. 319); or "a series of individual steps or instructions to be executed by the computer" (Ng & Cui, 2021, p. 6)	A sequence of steps set up according to mathematical rules or logic to perform a particular instruction for solving a mathematical task, e.g., to program a sequence of codes for testing prime number and then illuminating the LED lights accordingly (Ng & Cui, 2021); to move the robot based on a target geometric pattern (Jurado et al., 2020; Muñoz et al., 2020)	Rodríguez-Martínez et al. (2020); Sáez-López et al. (2019)
Events (handling)	"Handle one thing that causes another to happen" (Brennan & Resnick, 2012, as cited in Ng & Cui, 2021, p. 6); and "Instructions that make it possible to interact with objects in the programming environment" (Rodríguez-Martínez et al., 2020, p.319)	When an instruction that can be converted into mathematical information is executed, another instruction related to it with numerical (e.g., as pressing a button the variable "day" increases by 1, and another variable "the balance" will increase by 222; Ng & Cui, 2021) is executed	Rodríguez-Martínez et al. (2020); Sáez-López et al. (2019)
Subroutines	"A procedure that can be called within another procedure" (Sinclair & Patterson, 2018, p. 68)	A set of instructions designed to perform a specific mathematical procedure or task (e.g., geometric construction; Sinclair & Patterson, 2018) that can be used repeatedly within a program	None
Operators	"Provide support for mathematical, logical, and string expressions" (Brennan & Resnick, 2012, as cited in Ng & Cui, 2021, p. 6)	The of use of symbols or functions to denote or perform a mathematical (e.g., >, <, mod, ceiling, floor) or logical operation (e.g., AND, OR; Ng & Cui, 2021)	None

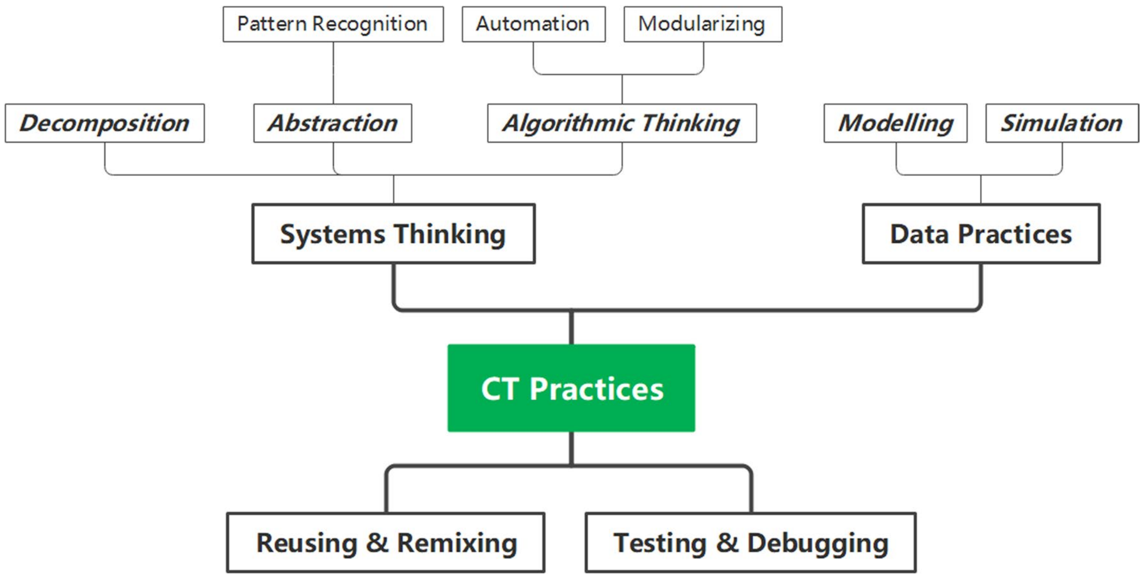


Fig. 6 Hierarchal relationship of computational practices in CT-based mathematics activities

manifestation of variables provides the empirical bases for future research to learn and expand on the diverse ways taken up by the use of variables in CT-based mathematics activities.

In addition, we highlight two other important CT concepts, *conditionals* and *loops*. A conditional is a statement needed to be tested in a program, upon which further actions would take place. A student's use of conditionals often requires their knowledge of number properties (e.g., remainder of a division is non-zero; Ng & Cui, 2021), algebraic properties (e.g., when a parametric equation determines the amount of rotation of the wheels; Ng & Cui, 2021; Sinclair & Patterson, 2018) or geometric properties (e.g., when a draggable point determines how much a given shape is scaled by; Sinclair & Patterson, 2018). In particular, Sinclair and Patterson (2018) noted the frequent applications of conditionals in conjunction with the signum function, $\text{sgn}(x)=\{-1, 0, 1\}$ to create conditions in a geometrized programming context, which "worked as a checkpoint of sorts, operating as an if–then statement in a block of computer code" (p. 68). A loop is a code structure that enables a repetition of a certain mathematical procedure. In the case of Miller's (2019) study, the students formulated mathematical procedures to draw a square using Scratch. This involved considering what actions (e.g., moving and turning) were repeated, how they were repeated (e.g., constant distance and angle), and how many times to repeat (e.g., 4 times). Furthermore, Sinclair and Patterson (2018) conceptualized the concept of loops as being closely related to the notions of iteration and infinity. They considered a triangle construction by joining the midpoints of the line segments which formed the triangle in the previous iteration. This process of constructing line segments between the midpoints (of the previous iteration) and generating new midpoints (of the current iteration) would yield an infinite number of triangles whose area would approach zero. Other researchers have considered the productive relationship between iterations and infinity in CT-based mathematical activities (Arnon et al., 2014; Weller et al., 2003); however, limited mathematical topics have been studied regarding the interplay between the concepts of infinity and loops in K-12 contexts. We suggest that there is high potential for CT-based mathematics tasks to take up these concepts in topics, such as sequences and series with infinite terms as well as fractal geometry.

Computational practices In terms of CT practices, we found that researchers have adopted diverse constructs and frameworks from extant literature (Bocconi et al., 2016; Brennan & Resnick, 2012; Hoyles & Noss, 2015; Weintrop et al., 2016), and there has been a lack of consensus about what each practice means and their relationship with each other. Here, we synthesize and organize these CT practices into different levels (see Fig. 6), hoping to offer a coherent and systematic understanding of how these CT practices manifest during the course of problem solving in CT-based mathematical activities.

First, we identify *systems thinking* to be one of the CT practices at the first level. Systems thinking entails analyzing the problem systematically by investigating a complex system as a whole, understanding the relationships within a system, thinking in levels, communicating information about a system, defining systems and managing complexity (Weintrop et al., 2016). We further categorize systems thinking into three sub-practices: *decomposition*, *abstraction*, and *algorithmic thinking*.

Decomposition refers to breaking down a complex problem into many small problems; it can also be applied in a mathematical sense of "decomposing numbers, shapes, or problems in mathematics" by "considering the part-whole relationships" (Shumway et al., 2021, p. 4). As illustrated in Pei et al.'s (2018) study, a student named Alice attempted to solve a problem of finding the area of an irregular polygon by breaking it down into sub-areas of triangles and rectangles. Besides decomposing a complex problem into smaller, manageable sub-problems, *abstraction* in the sense of abstracting key information and identifying patterns of the problem can also facilitate a solution process. A typical example of this practice is the reasoning process called *pattern recognition*, that is, looking for similarities between and within problems, e.g., to "see patterns, recognis[e] the unit of repeat, deduce the pattern and abstract the general structure" (Miller, 2019, p. 924). Pei and colleague's (2018) study demonstrated pattern recognition between problems; they described a group of students abstracting a general mathematical formula from a set of data, so that they could calculate the area of any lattice polygon regardless of its shape or size.

If decomposition and abstraction serve to simplify a complex problem at the thinking level, then algorithmic thinking serves to enact the simplification at the practice level through designing a concise algorithm to solve the problem. The reviewed studies commonly defined algorithmic thinking as the systematic means of articulating a problem's solution by well-defined rules, with logical, ordered, and discrete steps to be executed by a processing agent (Kaufmann & Stenseth, 2021; Ng & Cui, 2021; Valovičová et al., 2020). An important feature of algorithmic thinking is *automation*, which offers efficiency in problem solving by introducing automatic steps to be taken (Valovičová et al., 2020). After completing the algorithmic design for each sub-problem, it is necessary to combine these small parts to form a more complex algorithm, which is called *modularizing* (Ng & Cui, 2021).

While systems thinking has mainly to do with problem solving in general, its relevance to mathematical content is not particularly apparent. Our review identifies another first-level CT practice, *data practices*, with stronger mathematical connections, since they relate to collecting, creating, manipulating, analyzing, and visualizing data (Weintrop et al., 2016). Under data practices, we further characterized two inter-related practices, *simulation* and *modelling* (Dickes et al., 2020; Gilchrist et al., 2021). Often, CT-based mathematical problems require one to construct a model, represent the data from an experiment (e.g., simulating the spread of a disease; Gilchrist et al., 2021; modeling the calculation of a footprint problem; Dickes et al., 2020), and predict the future trend of the simulated problem (e.g., address problems related to disease transmission through the generated computational model; Gilchrist et al., 2021; generated and discussed the modeling graphs, made predictions related to footprint size using ViMAP's grapher; Dickes et al., 2020). Besides, data practices can serve the purpose of learning a mathematical concept during inquiry-based learning. In Pei et al.'s (2018) study, students used microworld to explore and derive the Pick's Theorem through gathering data, and recording, sorting, and examining the data set to look for and generalize relationships. This study was an example in which "students used computational tools to facilitate these data-driven computational thinking practices" (p. 85).

The remaining two categories of practices (i.e., Brennan & Resnick's [2012] *reusing and remixing, testing and debugging*) are specific to computational problem solving, in the sense that mathematical problem solving is necessarily situated in the programming environment and tool use. *Reusing and remixing* involve the coders building on and extending some existing codes and ideas to create a new or more complex program (Ng & Cui, 2021). *Testing and debugging* are necessary to ensure the solution can solve the tasks successfully (Cui & Ng, 2021). Testing often occurs before debugging, which is to "ensure that things work" (Ng & Cui, 2021, p. 6). If not, then debugging will follow to recognize, locate and fix errors or mistakes when they arise (Bouck & Yadav, 2020; Ng & Cui, 2021).

To summarize, our contribution is toward a systematic organization of the set of computational practices evident in the literature. It is worth noting that these practices can occur simultaneously to facilitate problem solving; meanwhile, they are the very CT learning outcomes co-developed with mathematics learning outcomes during the course of problem solving. An illustrative example was given by Sinclair and Patterson (2018), who discussed a variety of other CT concepts and practices evident when students constructed a complex machine in *Sketchpad* (see an example in Fig. 5). First, they decomposed the machine into components and actions into smaller movements and animations (*decomposition*). When creating small animations, they needed to understand the sequence of the steps to link them together into a complete operation (*algorithmic thinking*). To create these decomposed steps, students also needed to transform real phenomena (e.g., to create a parabolic trajectory) into mathematical objects (e.g., trigonometric or

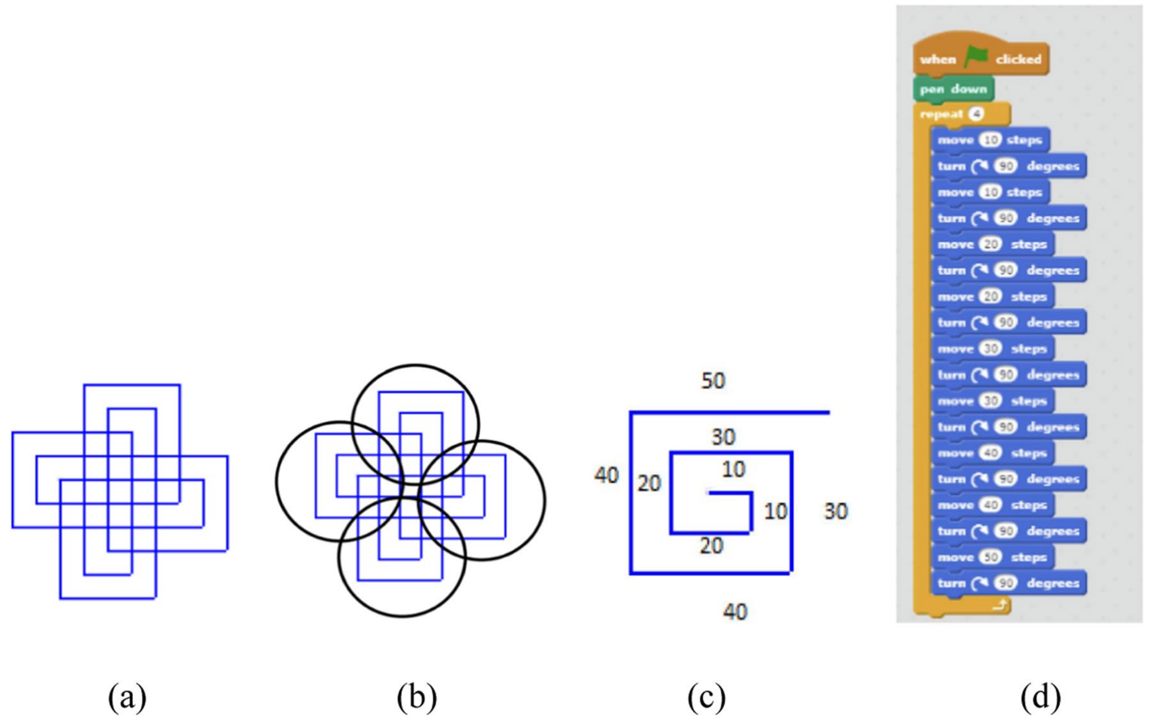
Table 5 Literature counts of each mathematical domain at different education levels

Education level	Content Domain (literature counts)	Sample topics and literature
Early Childhood & Primary Education	Numbers & Algebra (10)	- Number concepts and counting (Muñoz et al., 2020; Shumway et al., 2021) - Arithmetic operation and number line estimation (Sung et al., 2017) - Prime and composite number (Cui & Ng, 2021; Ng & Cui, 2021) - Whole numbers and negative numbers (Sáez-López et al., 2019) - Number sequence (Cui & Ng, 2021; Ng & Cui, 2021) - The greatest common divisor and the least common multiple (Rodríguez-Martínez et al., 2020) - Measurement of distance (Dickes et al., 2020; Valovičová et al., 2020) - Approximation (Dickes et al., 2020) - The steepest descent algorithm (Araya, 2021)
	Geometry (9)	- Coordinates (Sáez-López et al., 2019) - Spatial location (Muñoz et al., 2020; Shumway et al., 2021) - Length and angle (Bouck & Yadav, 2020) - Geometric figures (Jurado et al., 2020; Nogueira et al., 2021) - Solid figures (Echeverria et al., 2019) - Structures of fractals (Wilkerson-Jerde, 2014) - Properties of squares (Miller, 2019)
	Statistics & Probability (1)	- Data visualization: data ordering and manipulation; Creation and interpretation of graphics; Obtaining and interpreting information through data (Nogueira et al., 2021)
Secondary Education	Numbers & Algebra (4)	- Averaging, addition, and multiplication (Gilchrist et al., 2021) - Similarity, ratio, and measurement (Valovičová et al., 2020) - Parametric equations, functions, and trigonometry (Sinclair & Patterson, 2018) - Number sequence (Chan et al., 2021)
	Geometry (6)	- Properties of circles (Kaufmann & Stenseth, 2021) - Structures of fractals (Wilkerson-Jerde, 2014) - Shapes and area of triangles (Pei et al., 2018) - Segments and circles (Sinclair & Patterson, 2018) - Length and angle (Bouck & Yadav, 2020; Grizioti & Kynigos, 2021)
	Statistics & Probability (2)	- Data practices: collect, create, manipulate, analyze, and visualize data related to infectious pathogens (Gilchrist et al., 2021) - Solve a mathematical–statistical problem (Psychairs & Kallia, 2017)
	Calculus (0)	N/A

quadratic curve) (*abstraction and pattern recognition*). There were also multiple CT concepts involved in the activity: algebraic or geometric parameters that determine the characteristics of a machine (*variable*), the use of functions as conditions to control certain motions (*conditionals*), and the concept of *subroutines* and *loop* were used to operate programs controlled by certain buttons and to repeat certain actions. The vast examples provided in this section show that CT concepts and practices can fruitfully support mathematical meaning-making and representations of mathematical relationships when situated in the task of constructing a meaningful CT artefact. Thus, mathematical explorations can serve as a site for students to engage in CT in explicit ways, where the mathematical operations change what appear on the interface.

Mathematical learning outcomes demonstrated in computational contexts

Besides the aforementioned CT learning outcomes, the reviewed literature also conveys a range of mathematical ideas covered in CT-based mathematics instruction (see a summary in Table 5). Several studies have also researched the associated mathematical processes, such as problem-solving (Gilchrist et al., 2021; Psycharis & Kallia, 2017), algebraic thinking (Ng & Cui, 2021), modeling (Araya, 2021; Sinclair & Patterson, 2018), mathematical reasoning (Kaufmann & Stenseth, 2021), and generalization and pattern recognition (Miller, 2019; Wilerson-Jerde's, 2014). To further demonstrate how students' mathematical reasoning facilitates and emerges from computational contexts, we focus our discussion on three aspects. First, we discuss how students apply their mathematical knowledge to construct CT artefacts, including the challenges they may experience during the application. Next, we discuss how they draw upon their mathematics to anticipate and interpret CT outputs and how these processes are tied to their debugging practices. Then, we discuss the reciprocal influence in the sense that students' engagement with CT-math integrated tasks are generative to growth in their mathematical knowledge, and we identify ways through which such learning occurs.



```
when [flag] clicked
pen down
repeat (4)
    move (10) steps
    turn ↻ (90) degrees
    move (10) steps
    turn ↻ (90) degrees
    move (20) steps
    turn ↻ (90) degrees
    move (20) steps
    turn ↻ (90) degrees
    move (30) steps
    turn ↻ (90) degrees
    move (30) steps
    turn ↻ (90) degrees
    move (40) steps
    turn ↻ (90) degrees
    move (40) steps
    turn ↻ (90) degrees
    move (50) steps
    turn ↻ (90) degrees
```

Fig. 7 **a** Spirolateral shown to students, **b**, **c** students' conceived structure of the spirolateral, and **d** sample Scratch codes (Miller, 2019, pp. 923–924)

Applying mathematical knowledge to construct CT artefacts Numerous researchers have stressed the important role of mathematical knowledge in solving CT-based mathematical tasks, and in particular, in constructing computer programs to produce certain mathematical outcomes (e.g., Bouck & Yadav, 2020; Dickes et al., 2020; Grizioti & Kynigos, 2021; Miller, 2019). Influenced by Papert's (1980) *Logo* programming, several studies involved students constructing computer programs to generate a certain geometric shape (e.g., right triangles, squares, rectangles, parallelograms, trapezoids, and rhombuses; Bouck & Yadav, 2020; Miller, 2019). In doing so, the students necessarily drew on their knowledge of the constituting elements of the shapes (e.g., number of sides and angle size) and the relationship of those elements (e.g., parallelism or perpendicularity). A similar task design has been extended to the context of atypical shapes. For example, in Miller's (2019) study, a group of second graders (aged 7–8 years) were asked to use *Scratch* to draw a spirolateral (see Fig. 7a). The students were able to identify the structure of the spirolateral which pertained growing and repeating patterns (Fig. 7b, c), and then, they translated their observed structure into *Scratch* codes (Fig. 7d).

In algebraic contexts, a study was conducted by Dickes et al. (2020), in which a group of third graders used the command "repeat" to model the relationship between total distance and step size (Total Distance = No. of Repeats × Step size); this reflected that the students had constructed a multiplicative relationship between the quantities involved. As another example, Ng and Cui (2021) designed the Doubling Machine problem in which a bank account balance increased by a doubling amount of deposit in each subsequent week (i.e., \$3, \$9, \$21, \$45…). One pair of students observed that the next number always equals to the previous number times two plus three (i.e., $t_n = 2t_n + 3$) and translated this relationship into codes "[set button_value to][button_value * 2]" followed by "[set button_value to][button_value + 3]." In each of these examples, mathematics was used in an applied manner, and the students translated their conceived mathematical relationships and structures to computer languages and programs.

Despite the supporting role of mathematics in the construction of CT artefacts, there have been studies focused on the challenges students experience during their enactment of mathematics in computational contexts. As argued by Ng and colleagues (2021), these challenges were due to the mismatches or contradictions between CT and mathematical thinking (Cui & Ng, 2021; Ng et al., 2021; Sinclair & Patterson, 2018). For example, displaying a result requires an additional code during programming (e.g., calculating 5 + 3 = 8 and then displaying 8), whereas in mathematics, carrying out an operation automatically leads to a result (e.g., calculating 5 + 3 = 8 is a single operation; Cui & Ng, 2021; Ng et al., 2021). In addition, Jackiw and Finzer (1993) explained how debugging a sketch in dynamic geometry environments might differ from debugging in traditional programming environments. That is, the former requires "reasoning from the manifestation of a bug backwards…towards its cause" (ibid, p. 301), while the latter requires identifying the manifestation of a bug, locating the source of a bug, and re-executing the program circuitously.

The notions of variables and the way they are operated on also differ in mathematics and computing contexts. For example, in mathematics, the use of a variable in an equation already implies that the variable can take on any values, but in programming environments, one needs to write a line of codes to tell the computer to increase the variable by one (Cui & Ng, 2021). Another issue relates to the use of equal signs. In mathematics, one way of interpreting an equation is that two entities on both sides are equal (e.g., $y = x * 2$), but in programming, it means "performing some repetitive actions on a single quantity" (e.g., Set [variable] to [variable] *2 or [variable] = [variable]*2) (Cui & Ng, 2021, p. 19). As a result, students often encountered difficulties translating their conceived mathematical relationship from a problem situation into its programming equivalent.

To conclude, we highlight the take-away message by these empirical findings: despite the fact that mathematical knowledge plays a supportive role in constructing CT artefacts, the differences between CT and mathematical thinking may constrain students from understanding or practicing specific CT or mathematical concepts. As suggested by many scholars, these discrepancies might lead to problems in programming even if the students understand the problem situation and the mathematical relationships behind. Therefore, it is crucial for mathematics education researchers and mathematics educators to address these discrepancies to improve the integration of CT into mathematics learning.

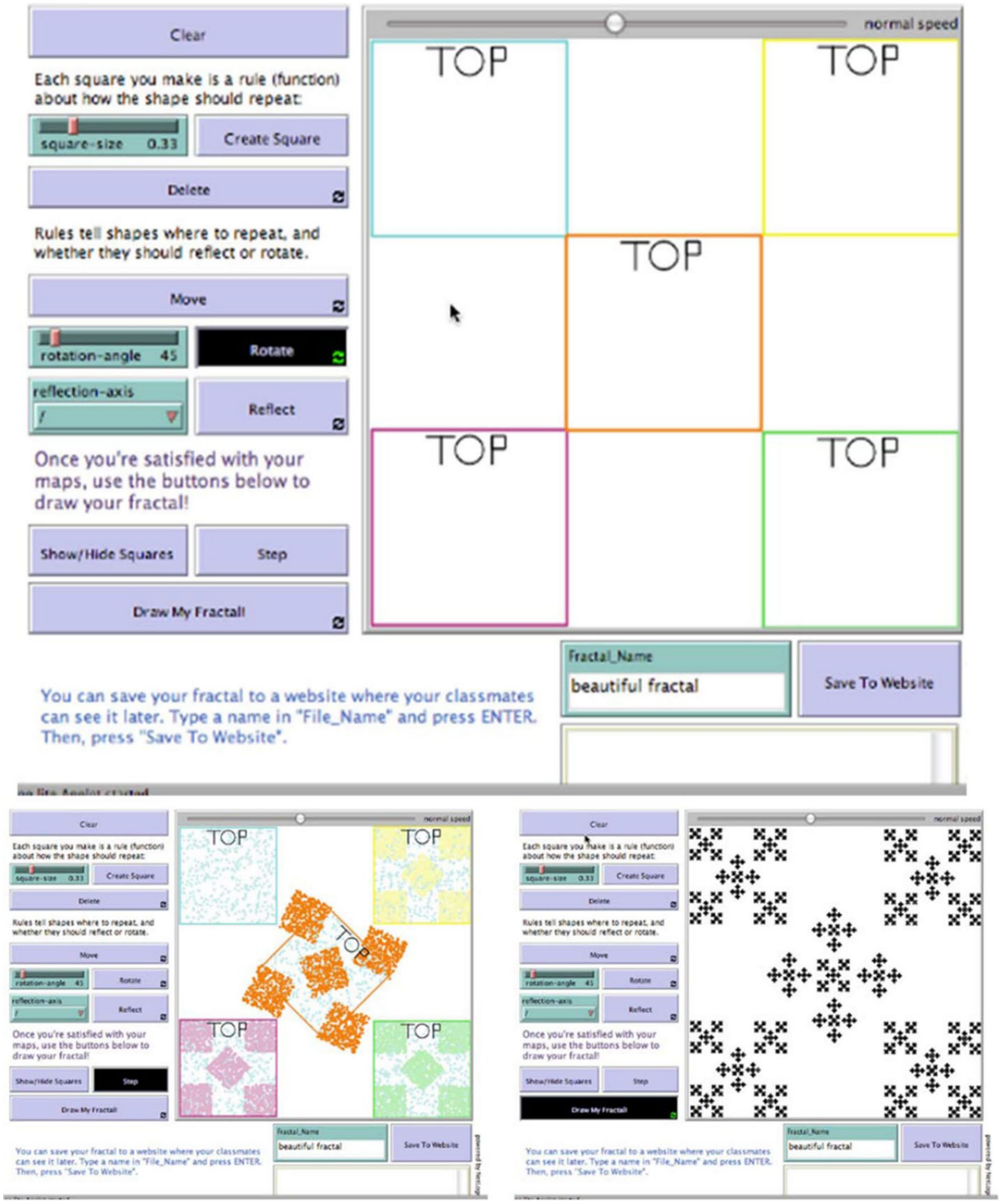
Applying mathematical knowledge to anticipate and interpret CT artefacts During programming, students may apply their mathematical knowledge to anticipate and interpret CT artefacts beyond making a workable program. We further classify four forms of such application based on the literature: (1) making sense of outputs, (2) anticipating outputs, (3) mediating debugging, and (4) code modeling.

Making sense of outputs One form of such application is applying one's mathematical knowledge to make sense of the outputs. In Wilerson-Jerde's (2014) study, a group of middle graders used a fractal construction tool (Fig. 8) to construct a collection of fractals. In doing so, the students interpreted why the produced fractals looked the way they were based on the creation rules (e.g., square-size, rotation-angle, reflection-axis). In a prior lesson, the students had already learned about fractals, including different types of fractals and how they could be produced. Therefore, the aim of the current activity was in supporting students' reflection upon features of fractals and in making sense of the rules that produce them in a CT context.

Another study by Pei et al. (2018) reported on a student making connections between her exploration within the *Lattice Land* environment and the triangle area formula she had previously learned. By interacting with *Lattice Land*, the student could see how moving a vertex along the line parallel to the base would change the shape of the triangle yet keep the area fixed. At first, the student was surprised by this observation, but then she recalled the area formula for a triangle ($Area = \frac{1}{2} \times Base \times Height$), which explained the conservation of area in the geometrized CT environment. Similar to the previous theme, the reasoning processes entailed by these two examples involve students' enactment of existing mathematical knowledge; what distinguish them from the prior theme is that the students were building novel connections between their mathematical knowledge and the CT outputs. Therefore, mathematics is not merely used in an applied manner, and the students deepen and reorganize their mathematical knowledge as they recontextualize it in a CT environment.

Anticipating outputs Studies observing this theme relate to the way students draw on their mathematical knowledge to anticipate the outputs of a computer program. The students are aware of the mathematics needed to evaluate the program and they compare its outputs to the intended goal. In Shumway et al.'s (2021) study, a group of kindergarten students (aged 5–6 years) engaged in an activity with the *Cubetto* robot coding toy, and they were asked to create programs to instruct the robot to move between two destinations. The robot's movement provided students instant feedback about the accuracy of the program in terms of the desired movement. When the robot moved some steps short of or over the goal, the students would revise the program by adding or removing some codes. This evidences how the students' knowledge of space, measurement, and number operations facilitated their anticipation of the robot movement and the associated debugging actions. As Shumway et al. (2021) described, students can "[think] one step ahead of the robot in anticipation of evaluating the accuracy of their sequence of codes in the program" (p. 20).

Dickes et al. (2020) offered another example of students realizing their program was incorrect due to their outputs being inconsistent with their mathematical



Each square you make is a rule (function) about how the shape should repeat:

square-size 0.33

Rules tell shapes where to repeat, and whether they should reflect or rotate.

rotation-angle 45

reflection-axis /

Once you're satisfied with your maps, use the buttons below to draw your fractal!

Fractal_Name: beautiful fractal

You can save your fractal to a website where your classmates can see it later. Type a name in "File_Name" and press ENTER. Then, press "Save To Website".

Fig. 8 Fractal construction interface and resulting computational objects (Wilkerson-Jerde, 2014, p. 108)

calculations. Specifically, the students noticed that the total distance represented in their graph was 240 units, while the total distance calculated using the rate equation was 300 units (15 step size × 20 number of steps = 300 units of total distance). They subsequently revisited the program and corrected the programming error to obtain a desired result. Similar to the previous example, the students' mathematical thinking guided their evaluation of the computer program and motivated their debugging.

Mediating debugging practices We also identify examples, where mathematical reasoning not only *motivated* students' debugging practices but also *guided and mediated* the debugging practices themselves. In a study by

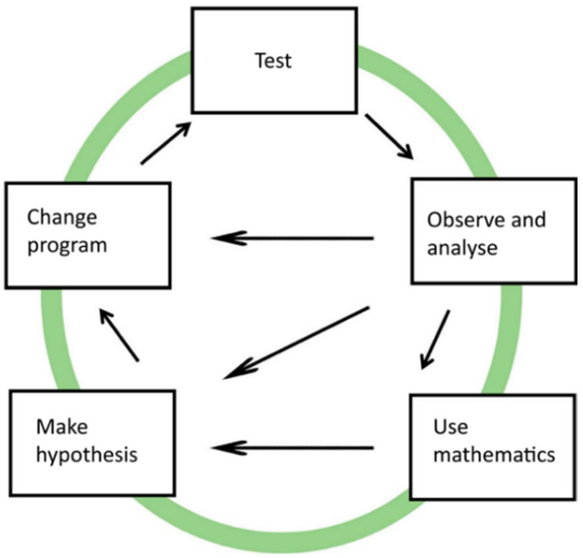


Fig. 9 Iteration cycle (Kaufmann & Stenseth, 2021, p. 1043)

Kaufmann and Stenseth (2021), a group of students in Grade 8 and Grade 9 (age 13 and 14) were asked to troubleshoot a program, so that a wheel would rotate along a line, such that its diameter hits the target markers on the line. The students initially used intuition and trial-and-error to change their codes without any mathematical basis; eventually, they attended to the circumference and its relationship to other quantities to debug the program successfully. In this example, mathematics was not applied to construct the program per se, but it was used for locating and fixing the bugs. Kaufmann and Stenseth (2021) further proposed a model to characterize how mathematics mediates the complex process of troubleshooting; as they shown in Fig. 9, the use of mathematics is intertwined within the iterative process of testing, observing and analyzing, making hypothesis, and changing program.

Code modeling Code modeling refers to students interpreting the mathematics underlying a sequence of codes, sometimes during the process of debugging. Ng and Cui (2021) described a pair of students who debugged a program for generating a prime number detector. They modeled the outcomes of two set of codes, with one having the conditions ('remainder = 0' and 'remainder > 0') within a repeat loop and one outside the loop, and they realized that they should have placed the conditions within the loop to check the remainder before (as opposed to after) carrying out the next division. Different from the previous examples, this instance is illustrative of students using their mathematical knowledge to model the behavior of a given set of codes. *Code modeling* allowed them to realize that the codes behaved in a way against the intended mathematical operations, and hence to revise the program accordingly.

Summary Examples in these four themes commonly involve students applying their mathematical knowledge to anticipate or interpret CT artefacts, but there are nuances between them. First, using mathematics to make sense of CT outputs can engender novel mathematical connections that were not present previously. In terms of the other three themes which relate directly to debugging practices, students can (1) recognize a buggy program by comparing the outputs with the anticipated outcomes; (2) fix the bugs by reconsidering the mathematical relationships relevant to the situation; or (3) fix the bugs by modeling the mathematical behavior underlying the program. We consider it significant to tease out and illustrate these nuances across the literature, since they shed light on the foundational role of mathematics for CT as well as the insights into how mathematics is used in explicit but different ways in CT contexts.

Generating new mathematical knowledge in parallel with CT development Engaging with CT-based mathematical tasks not only requires application of mathematical knowledge but also in turn contributes to the emergence of novel mathematical knowledge. A number of quantitative studies have demonstrated positive correlation between students' learning of CT and learning of mathematics (Echeverría et al., 2019; Nogueira et al., 2021; Rodríguez-Martínez et al., 2020; Sáez-López et al., 2019; Sung et al., 2017). Echoing the quantitative findings, researchers adopting qualitative methods have confirmed the positive influence of students' CT experiences on mathematical development, showing evidence of how mathematical learning unfolds in CT contexts. We identify three avenues through which new mathematical knowledge emerges in CT-based mathematics activities.

First, students can construct mathematical ideas and relationships by reflecting on CT outputs. For example, in Wilerson-Jerde's (2014) study, the students were asked to use the *Categorizer* tool to create categories that reflected certain features about fractals (e.g., density, "crispness" or "fuzziness", twisting structure, branching structure, sponging structure, and area reduction). By observing a collection of fractals produced by the computer, the students were offered opportunities to "explore important mathematical properties of those structures, and offered more ways to construct fractals with particular mathematical properties" (p. 118). In another study by Pei et al. (2018), students collected data from a number

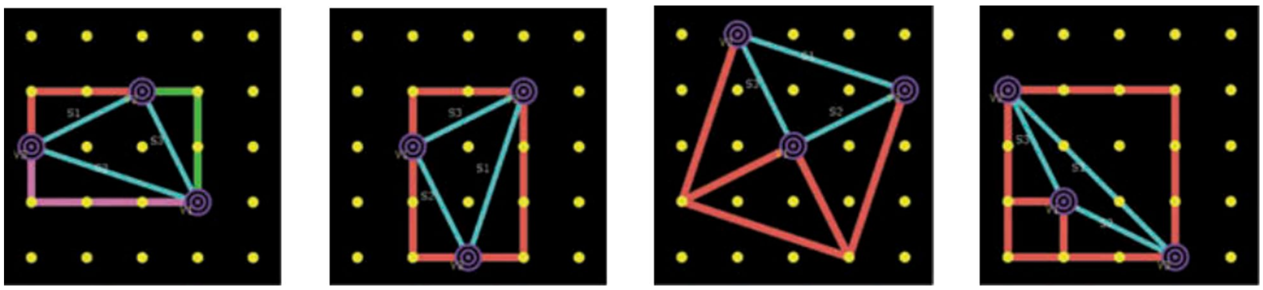


Fig. 10 Student discovered strategies for calculating the area of a triangle in Lattice Land (Pei et al., 2018, p. 81)

of polygons in *Lattice Land* and derived the Pick's Theorem[1] by reasoning with the data and generalizing pattern from the data. In each of these examples, students made use of the affordances of computational tools to create numerous examples and outcomes that they could reflect upon and generalize from. The rich mathematical conversations around those CT outputs facilitated the students' constructions of deep mathematical ideas and relationships.

Another avenue through which new mathematical ideas are constructed is reflecting on the programming processes and code features. In Miller's (2019) study, a group of students created codes for making a square and deduced a generalized rule for calculating the perimeter of a square by identifying the repeating patterns from the codes (i.e., perimeter of a square equals four times its length). The author concluded that these primary school students "were forming generalizations when writing computer codes" (ibid, p. 925), suggesting that teaching mathematics through coding may provide opportunities for students to "see the mathematics in a deeper way and build a more connected understanding" (ibid, p. 923). As another example, Pei et al. (2018) described a classroom discussion centered on calculating the areas of triangles whose side lengths were not whole numbers in the *Lattice Land* environment (Fig. 10). Using the "draw segment" tool, the students came up with different ways to "box" the triangles and calculate their areas. These novel methods were only possible when students used the construction function of the programming tool and reflected on the construction processes and products.

Third, CT-based mathematics instruction motivates a new form of mathematical knowledge that is represented by computational languages and unique to CT contexts. For example, Ng and Cui (2021) discussed a group of elementary students who had no prior knowledge of algebraic expressions and variables, but they could use iterations like '[set total to] [total + money]' and '[set money to] [money *2]' to model a geometric series. It was noteworthy that these students could make use of commands and codes to model mathematical relationships whose algebraic forms were not yet accessible to them in paper-and-pencil context. Therefore, the authors concluded that mathematical content can be "purposefully synergized in a programming context" (ibid, p. 859) to make advanced and historically challenging ideas accessible to younger children and "can serve as a bridge for children to advance their arithmetic thinking toward building coherent and meaningful learning of more advanced algebra in later years" (ibid, p. 858). Shumway et al. (2021) made a similar claim by showing evidence of kindergarten students expressing their emerging knowledge of spatial orientation, spatial movement, measurement, and numbers by imitating a robot toy's motion, suggesting their mathematical knowledge is embedded in their programming activities. The author also argued that those activities afforded students coordinating interconnected domains of mathematical knowledge (i.e., simultaneously using their spatial reasoning and knowledge of numbers) and constructing dynamic perspective of relevant concepts, which went beyond the kindergarten standards.

In summary, this group of literature provides rich evidence that confirms the affordances of CT-math integration in K-12 education. Such integration supports students' abstraction of a variety of mathematical ideas (e.g., geometric properties, formulas, theorems). Through interacting with the CT environment, students are offered opportunities to explore a variety of mathematical contents through reflecting upon the CT construction processes and products. Furthermore, the CT-math integration opens up new possibilities of learning mathematics, ultimately transforming traditional views of the nature of mathematics and mathematics learning. As the literature showed, the integration increases accessibility

[1] The Pick's Theorem states that suppose that a polygon has integer coordinates for all of its vertices. Let I be the number of integer points that are interior to the polygon, and let B be the number of integer points on its boundary (including vertices as well as points along the sides of the polygon). Then, the area A of the polygon is: $A = 0.5B + I - 1$.

of mathematical ideas to young kids and allows more mathematical connections made. It also contributes to the emergence of a new form of mathematical knowledge and reasoning mode that bear characteristics of computing (e.g., syntax representations of algebraic relationships), which leave seeds for students' future mathematical abstraction and development.

Discussion and future research directions

CT is the thought process entailing abstracting, algorithmic thinking, and computational modeling in combination with the use of programming tools to solve a series of complex problems (Wing, 2006). When engaging CT-based mathematical activities, students not only produce CT artefacts and programmable solutions in a physical sense, but also construct meanings for various CT and mathematical concepts. Despite positive findings about the impact of CT-based mathematical activities on student learning, consensus has not been reached regarding how to integrate the two fields in coherent and meaningful ways. This systematic review has clarified, with concrete examples, the topics, education levels, pedagogical arrangements including learning resources, and the degree of integration among other STEM disciplines in which recent studies have investigated in relation to CT-based mathematics learning. Together, these studies have shown that no matter which descriptor of learning context is concerned, CT should not be introduced as an add-on to mathematics learning, but considerations for research and practice is warranted at a conceptual level to fully understand and maximize the ways CT may support mathematics learning.

Our study sets out to review the impact of CT-based mathematics instruction on student learning through four aspects: educational context, pedagogical approach, theoretical construct related to CT and mathematics learning outcomes, and the interplay between CT and mathematics in CT-based mathematical activities. From each of these aspects, our review suggests several affordances of CT which can foster and enrich mathematics learning: first, we observe that programming activities afford non-prescribed mathematics learning in unstructured ways and with little prior knowledge necessary. Second, problem-based learning and project-based learning are particularly compatible with CT-based mathematics instruction. Third, CT integration in mathematics education has proven to be supportive in the content of algebra and geometry across education levels. Finally, CT can serve as both the means (when applying mathematical knowledge to anticipate or interpret CT artefacts) and goals (when applying mathematical knowledge to construct CT artefacts) of learning in CT-based mathematics activities. That is, CT-based mathematical activities afforded students to develop and apply their CT alongside their mathematical knowledge. In particular, we evidenced students' CT development by their ability to use diverse commands, flexibility in sequencing codes, and efficiency in debugging (Dickes et al., 2020; Kaufmann & Stenseth, 2021; Ng & Cui, 2021); simultaneously as the students' CT concepts and practices emerged, the CT-based mathematics activities also became progressively more mathematically grounded and rigorous.

Altogether, these characteristics have important implications to how we organize mathematics learning activities in computationally enhanced ways, and if taken up, can be highly transformative in the way we structure the mathematics curricula and instruction. Wilkerson-Jerde (2014) posits that, "computational ideas provide new and powerful ways of thinking about math and science phenomena" (p. 101); specifically, we highlight the re-orientation of sequencing of mathematical topics as well as the pedagogical aspects of mathematics instructions that are made possible in CT environments (Ng et al., forthcoming). In addition, we draw attention to the mutual support and affordance of CT and mathematical activities, and in turn, highlight the potential of designing CT instruction around mathematical inquiry for co-developing CT and mathematical understandings (Pei et al., 2018).

Overall, this systematic review corroborates with previous findings that computing reflexively develops alongside mathematical (and more broadly, STEM) practices (Dickes et al., 2020; Kafai & Harel, 1991). As Kaufmann and Stenseth (2021) asked at the end of the article: "Have we designed a problem that utilizes programming to give a better understanding of mathematics, or are we using mathematics to improve the problem-solving skills in programming" (p. 1045)? We propose that the answer is "both". That is, CT-based mathematics instruction entails an interactive and cyclical process of reasoning mathematically and reasoning computationally. Likewise, Sinclair and Patterson (2018) claim that CT and mathematics are closely overlapping and are parallel in many aspects. Pei et al. (2018) thus suggested blended curricula that make use of the close connection between CT and disciplinary learning, saying:

> *One way forward [...] is through the creation of blended curricula that infuse computational thinking practices into topics from other domains. In doing so, the destination domain can serve as a context for developing and using computational thinking practices while also presenting learners with new, computationally facilitated approaches for exploring disciplinary ideas. (p. 76)*

Besides, we conclude that research on the following areas is currently lacking:

1. CT-based mathematics instructions taking place in formal and interdisciplinary education settings. This also implies that teachers' professional development concerning the emergent competency of CT-based mathematics thinking is indispensable.
2. design experiments within geometrized programming environments, and more generally, the role of different computing tools in supporting mathematics learning;
3. empirical studies highlighting the different ways in which CT and mathematics learning co-develop, including the potential of integrating CT in other domains of mathematics (e.g., calculus and statistics), and what can be learned from CT-based mathematics instructions in these contexts, both in terms of how CT support mathematics learning, and vice versa.

In conclusion, this study lays the groundwork for future research into more coherent and effective integration of CT in K-12 mathematics education. We encourage more research on the integration of child-friendly programming instructional designs and tools for adoption in primary education or lower to begin integration of CT concepts early in mathematics teaching and learning. Regarding undergraduate mathematics education, we suggest that, due to the nature of the mathematical content, the integration of CT in this context seems to be more natural and common than K-12 education. Different than K-12 mathematics teaching that often rely on paper-and-pencil, it is not uncommon to find an undergraduate mathematics class that includes CT components. However, despite the predominate CT-math integration at the collegiate level, there has been few research that provides fine-grained analysis of the interplay between CT and mathematics learning, although recent work by Lockwood, Sand, and colleagues have paid explicit attention to this research line (Lockwood, 2022; Lockwood & De Chenne, 2020, 2021; Sand et al., 2022). In addition, more interestingly, these scholars' findings echo the types of interplay we have identified in the current review, and future research can continue to gain insights into how the interactions between CT and mathematical thinking may manifest differently or similarly in both contexts (Lockwood & Mørken, 2021; Lockwood, et al., 2019).

Notably, although most studies provide empirical evidence in the mutual relationship between CT and mathematics learning outcomes, authors of two articles conveyed that students' CT engagement did not support their mathematical learning (Chan et al., 2021; Psycharis & Kallia, 2017). Given the contradictory claims of these two studies with other studies reviewed, there remains future work in understanding how to maximize the effective role of CT in students' mathematical learning and vice versa. Specifically, to offer an explanation for these discrepant results, researchers need to not only focus on students' learning outcomes and their correlations with CT engagement, but also focus on the qualitative nuances in the students' learning processes and identify and compare the essential features of the tasks and instructional approaches that generate the discrepant learning outcomes.

Abbreviations

CT	Computational thinking
PRISMA	Preferred Reporting Items for Systematic Reviews and Meta-Analyses
STEM	Science, Technology, Engineering and Mathematics
STEAM	Science, Technology, Engineering, Arts and Mathematics
WOS	Web of Science

Acknowledgements
This paper was fully supported by the Hong Kong Research Grants Committee, General Research Fund (Ref No. 14603720).

Author contributions
All authors contributed to the design and implementation of this study. Literature searches with agreed-upon search terms were completed by the first author. Then, the first, second and third author collaboratively worked on the process of literature selection, coding, and themes identification. The first author and second author conducted the analysis and interpretation of the reviewed literature with help of the third and fourth author. Findings were discussed and agreed by all authors. All authors read and approved the final manuscript.

Funding
This study is supported by the Hong Kong Research Grants Committee, General Research Fund (Ref No. 14603720).

Availability of data and materials
This is a systematic review of prior literature. All articles we reviewed here are accessible in Web of Science database.

Declarations

Competing interests
There is no potential conflict of interest in the work reported here.

Received: 25 February 2022 Accepted: 2 January 2023
Published online: 18 January 2023

References

***Reference in the reviewed list**

*Araya, R. (2021). Enriching elementary school mathematical learning with the steepest descent algorithm. *Mathematics*. https://doi.org/10.3390/math9111197

Arnon, I., Cottrill, J., Dubinsky, E., Oktaç, A., Fuentes, S. R., Trigueros, M., & Weller, K. (2014). *APOS theory: A framework for research and curriculum development in mathematics education*. Springer.

Baldwin, D., Walker, H. M., & Henderson, P. B. (2013). The roles of mathematics in computer science. *ACM Inroads, 4*(4), 74–80. https://doi.org/10.1145/2537753.2537777

Barcelos, T. S., Munoz, R., Villarroel, R., Merino, E., & Silveira, I. F. (2018). Mathematics learning through computational thinking activities: A systematic literature review. *Journal of Universal Computer Science, 24*(7), 815–845.

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.

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

Blumenfeld, P. C., Soloway, E., Marx, R. W., Krajcik, J. S., Guzdial, M., & Palincsar, A. (1991). Motivating project-based learning: Sustaining the doing, supporting the learning. *Educational Psychologist, 26*(3–4), 369–398.

Bocconi, S., Chioccariello, A., Dettori, G., Ferrari, A., Engelhardt, K., Kampylis, P., & Punie, Y. (2016). *Developing computational thinking in compulsory education*. European Commission, JRC Science for Policy Report.

Bouck, E. C., Sands, P., Long, H., & Yadav, A. (2021). Preparing special education preservice teachers to teach computational thinking and computer science in mathematics. *Teacher Education and Special Education, 44*(3), 221–238. https://doi.org/10.1177/0888406421992376

*Bouck, E. C., & Yadav, A. (2020). Providing access and opportunity for computational thinking and computer science to support mathematics for students with disabilities. *Journal of Special Education Technology, 37*(1), 151–160.

Bråting, K., & Kilhamn, C. (2021). Exploring the intersection of algebraic and computational thinking. *Mathematical Thinking and Learning, 23*(2), 170–185.

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 annual meeting of the American educational research association* (pp. 1–25).

*Chan, S.-W., Looi, C.-K., Ho, W. K., Huang, W., Seow, P., & Wu, L. (2021). Learning number patterns through computational thinking activities: A Rasch model analysis. *Heliyon, 7*(9), e07922. https://doi.org/10.1016/j.heliyon.2021.e07922

*Cui, Z., & Ng, O. (2021). The interplay between mathematical and computational thinking in primary school students' mathematical problem-solving within a programming environment. *Journal of Educational Computing Research, 59*(5), 988–1012. https://doi.org/10.1177/0735633120979930

De Chenne, A., & Lockwood, E. (2022). A task to connect counting processes to lists of outcomes in combinatorics. *The Journal of Mathematical Behavior, 65*, 100932.

*Dickes, A. C., Farris, A. V., & Sengupta, P. (2020). Sociomathematical norms for integrating coding and modeling with elementary science: A dialogical approach. *Journal of Science Education and Technology, 29*(1), 35–52. https://doi.org/10.1007/s10956-019-09795-7

*Echeverria, L., Cobos, R., & Morales, M. (2019). Improving the students computational thinking skills with collaborative learning techniques. *Revista Iberoamericana De Tecnologias Del Aprendizaje, 14*(4), 196–206. https://doi.org/10.1109/RITA.2019.2952299

English, L. D. (2016). STEM education K-12: Perspectives on integration. *International Journal of STEM Education, 3*(1), 1–8.

Ernst, D. C., Hodge, A., & Yoshinobu, S. (2017). What is inquiry-based learning. *Notices of the AMS, 64*(6), 570–574.

Eshach, H. (2007). Bridging in-school and out-of-school learning: Formal, non-formal, and informal education. *Journal of Science Education and Technology, 16*(2), 171–190. https://doi.org/10.1007/s10956-006-9027-1

Feldhausen, R., Weese, J. L., & Bean, N. H. (2018). Increasing student self-efficacy in computational thinking via STEM outreach programs. In *Proceedings of the 49th ACM technical symposium on computer science education* (pp. 302–307). https://doi.org/10.1145/3159450.3159593

*Gilchrist, P. O., Alexander, A. B., Green, A. J., Sanders, F. E., Hooker, A. Q., & Reif, D. M. (2021). Development of a pandemic awareness stem outreach curriculum: Utilizing a computational thinking taxonomy framework. *Education Sciences*. https://doi.org/10.3390/educsci11030109

*Grizioti, M., & Kynigos, C. (2021). Code the mime: A 3D programmable charades game for computational thinking in MaLT2. *British Journal of Educational Technology, 52*(3), 1004–1023. https://doi.org/10.1111/bjet.13085

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

Hickmott, D., Prieto-Rodriguez, E., & Holmes, K. (2018). A scoping review of studies on computational thinking in K–12 mathematics classrooms. *Digital Experiences in Mathematics Education, 4*(1), 48–69. https://doi.org/10.1007/s40751-017-0038-8

Ho, W. K., Lool, C. K., Huang, W., Seow, P., & Wu, L. (2021). Computational thinking in mathematics: To be or not to be, that is the question. In *Mathematics—Connection and beyond: Yearbook 2020 association of mathematics educators* (pp. 205–234).

Hong Kong Curriculum Development Council. (2020). *Computational thinking–Coding education: Supplement to the primary curriculum*. The Printing Department.

Horn, M. S., & Jacob, R. J. (2006). Tangible programming in the classroom: a practical approach. *CHI'06 Extended Abstracts on Human Factors in Computing Systems*, 869–874.

Hoyles, C., & Noss, R. (2015). *Revisiting programming to enhance mathematics learning*. Presentation at Math+ Coding Symposium, Western University, London. Retrieved from https://researchideas.ca/coding/

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 and 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

Jackiw, N., & Finzer, W. (1993). The geometer's sketchpad: Programming by geometry. In A. Cypher (Ed.), *What what I do: Programming by demonstration* (pp. 293–307). The MIT Press.

Jocius, R., Ian O'byrne, W., Albert, J., Joshi, D., Robinson, R., & Andrews, A. (2021). Infusing computational thinking into STEM teaching. *Technology & Society, 24*(4), 166–179. https://doi.org/10.2307/48629253

*Jurado, E., Fonseca, D., Coderch, J., & Canaleta, X. (2020). Social steam learning at an early age with robotic platforms: A case study in four schools in Spain. *Sensors (Switzerland), 20*(13), 1–23. https://doi.org/10.3390/s20133698

Kafai, Y., & Harel, I. (1991). Children learning through consulting: When mathematical ideas, knowledge of programming and design, and playful discourse are intertwined. In I. Harel & S. Papert (Eds.), *Constructionism* (pp. 110–140). Ablex.

*Kaufmann, O. T., & Stenseth, B. (2021). Programming in mathematics education. *International Journal of Mathematical Education in Science and Technology, 52*(7), 1029–1048. https://doi.org/10.1080/0020739X.2020.1736349

Kotsopoulos, D., Floyd, L., Khan, S., Namukasa, I. K., Somanath, S., Weber, J., & Yiu, C. (2017). A pedagogical framework for computational thinking. *Digital Experiences in Mathematics Education, 3*(2), 154–171. https://doi.org/10.1007/s40751-017-0031-2

Lockwood, E. (2022). Leveraging prediction and reflection in a computational setting to enrich undergraduate students' combinatorial thinking. *Cognition and Instruction*. https://doi.org/10.1080/07370008.2021.2020793

Lockwood, E., & De Chenne, A. (2020). Enriching students' combinatorial reasoning through the use of loops and conditional statements in Python. *International Journal of Research in Undergraduate Mathematics Education, 6*(3), 303–346. https://doi.org/10.1007/S40753-019-00108-2

Lockwood, E., & De Chenne, A. (2021). Reinforcing key combinatorial ideas in a computational setting: A case of encoding outcomes in computer programming. *Journal of Mathematical Behavior, 62*, 100857. https://doi.org/10.1016/j.jmathb.2021.100857

Lockwood, E., De Jarnette, A. F., & Thomas, M. (2019). Computing as a mathematical disciplinary practice. *Journal of Mathematical Behavior, 54*, 100688. https://doi.org/10.1016/j.jmathb.2019.01.004

Lockwood, E., & Mørken, K. (2021). A call for research that explores relationships between computing and mathematical thinking and activity in RUME. *International Journal of Research in Undergraduate Mathematics Education, 7*(3), 404–416. https://doi.org/10.1007/s40753-020-00129-2

Matsumoto, P. S., & Cao, J. (2017). The development of computational thinking in a high school chemistry course. *Journal of Chemical Education, 94*(9), 1217–1224. https://doi.org/10.1021/acs.jchemed.6b00973

*Miller, J. (2019). STEM education in the primary years to support mathematical thinking: Using coding to identify mathematical structures and patterns. *ZDM - Mathematics Education, 51*(6), 915–927. https://doi.org/10.1007/s11858-019-01096-y

*Muñoz, L., Villarreal, V., Morales, I., Gonzalez, J., & Nielsen, M. (2020). Developing an interactive environment through the teaching of mathematics with small robots. *Sensors (Switzerland)*. https://doi.org/10.3390/s20071935

National Research Council. (2013). *Next generation science standards: For states, by states*. The National Academies Press.

*Ng, O., & Cui, Z. (2021). Examining primary students' mathematical problem-solving in a programming context: Towards computationally enhanced mathematics education. *ZDM - Mathematics Education, 53*(4), 847–860. https://doi.org/10.1007/s11858-020-01200-7

*Ng, O., Liu, M., & Cui, Z. (2021). Students' in-moment challenges and developing maker perspectives during problem-based digital making. *Journal of Research on Technology in Education, 4*, 1–15.

Ng, O., Sinclair, N., Ferrara, F., & Liang, B. (2023). Transforming arithmetic through digital resource. In B. Pepin, G. Gueudet, & J. Choppin (Eds.), *Handbook of digital (curriculum) resources in mathematics education*. Springer.

*Nogueira, V. B., Teixeira, D. G., de Lima, I. A. C. N., Moreira, M. V. C., de Oliveira, B. S. C., Pedrosa, I. M. B., de Queiroz, J. W., & Jeronimo, S. M. B. (2021). Towards an inclusive digital literacy: An experimental intervention study in a rural area of Brazil. *Education and Information Technologies, 27*(2), 2807–2834.

Page, M. J., McKenzie, J. E., Bossuyt, P. M., Boutron, I., Hoffmann, T. C., Mulrow, C. D., & Moher, D. (2021a). The PRISMA 2020 statement: An updated guideline for reporting systematic reviews. *Systematic Reviews, 10*, 89. https://doi.org/10.1186/s13643-021-01626-4.

Page, M. J., Moher, D., Bossuyt, P. M., Boutron, I., Hoffmann, T. C., Mulrow, C. D., & McKenzie, J. E. (2021b). PRISMA 2020 explanation and elaboration: Updated guidance and exemplars for reporting systematic reviews. *BMJ, 372*, n160. https://doi.org/10.1136/bmj.n160.

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

Papert, S., & Harel, I. (1991). Situating constructionism. *Constructionism, 36*(2), 1–11.

*Pei, C., Weintrop, D., & Wilensky, U. (2018). Cultivating computational thinking practices and mathematical habits of mind in Lattice Land. *Mathematical Thinking and Learning, 20*(1), 75–89. https://doi.org/10.1080/10986065.2018.1403543

*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

Rich, K. M., Spaepen, E., Strickland, C., & Moran, C. (2020). Synergies and differences in mathematical and computational thinking: Implications for integrated instruction. *Interactive Learning Environments, 28*(3), 272–283. https://doi.org/10.1080/10494820.2019.1612445

*Rodríguez-Martínez, J. A., González-Calero, J. A., & Sáez-López, J. M. (2020). Computational thinking and mathematics using Scratch: An experiment with sixth-grade students. *Interactive Learning Environments, 28*(3), 316–327. https://doi.org/10.1080/10494820.2019.1612448

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

*Sáez-López, J. M., Sevillano-García, M. L., & Vazquez-Cano, E. (2019). The effect of programming on primary school students' mathematical and scientific understanding: Educational use of mBot. *Educational Technology Research and Development, 67*(6), 1405–1425. https://doi.org/10.1007/s11423-019-09648-5

Sand, O. P., Lockwood, E., Caballero, M. D., & Mørken, K. (2022). Three cases that demonstrate how students connect the domains of mathematics and computing. *The Journal of Mathematical Behavior, 67*, 100955. https://doi.org/10.1016/j.jmathb.2022.100955

Savery, J. R. (2015). Overview of problem-based learning: Definitions and distinctions. *Essential Readings in Problem-Based Learning: Exploring and Extending the Legacy of Howard s. Barrows, 9*(2), 5–15.

*Shumway, J. F., Welch, L. E., Kozlowski, J. S., Clarke-Midura, J., & Lee, V. R. (2021). Kindergarten students' mathematics knowledge at work: the mathematics for programming robot toys. *Mathematical Thinking and Learning*. https://doi.org/10.1080/10986065.2021.1982666

*Sinclair, N., & Patterson, M. (2018). The dynamic geometrisation of computer programming. *Mathematical Thinking and Learning, 20*(1), 54–74. https://doi.org/10.1080/10986065.2018.1403541

Sırakaya, M., Alsancak Sırakaya, D., & Korkmaz, Ö. (2020). The impact of STEM attitude and thinking style on computational thinking determined via structural equation modeling. *Journal of Science Education and Technology, 29*(4), 561–572. https://doi.org/10.1007/s10956-020-09836-6

Sneider, C., Stephenson, C., Schafer, B., & Flick, L. (2014). Exploring the science framework and NGSS: Computational thinking in the science classroom. *Science Scope, 38*(3), 10–15.

Strauss, A., & Corbin, J. (1990). *Basics of qualitative research*. Sage Publications.

*Sung, W., Ahn, J., & Black, J. B. (2017). Introducing computational thinking to young learners: Practicing computational perspectives through embodiment in mathematics education. *Technology, Knowledge and Learning, 22*(3), 443–463. https://doi.org/10.1007/S10758-017-9328-X

Swaid, S. I. (2015). Bringing computational thinking to STEM education. *Procedia Manufacturing, 3*, 3657–3662. https://doi.org/10.1016/j.promfg.2015.07.761

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

Testa, J. (2009). The Thomson Reuters journal selection process. *Transnational Corporations Review, 1*(4), 59–66.

*Valovičová, L., Ondruška, J., Zelenický, L., Chytrý, V., & Medová, J. (2020). Enhancing computational thinking through interdisciplinary steam activities using tablets. *Mathematics, 8*(12), 1–15. https://doi.org/10.3390/math8122128

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

Weller, K., Clark, J., Dubinsky, E., Loch, S., McDonald, M., & Merkovsky, R. (2003). Student performance and attitudes in courses based on APOS theory and the ACE teaching cycle. In A. Selden, E. Dubinsky, G. Harel, F. Hitt (Eds.), *Research in collegiate mathematics education V* (pp. 97–131).

*Wilkerson-Jerde, M. H. (2014). Construction, categorization, and consensus: Student generated computational artifacts as a context for disciplinary reflection. *Educational Technology Research and Development, 62*(1), 99–121. https://doi.org/10.1007/s11423-013-9327-0

Wing, J. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33–35.

Wing, J. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society a: Mathematical, Physical and Engineering Sciences, 366*(1881), 3717–3725.

Wing, J. (2011). Research notebook: Computational thinking—What and why. *The Link Magazine, 6*, 20–23.

Publisher's Note