Digit Exp Math Educ (2018) 4:48–69
https://doi.org/10.1007/s40751-017-0038-8

MATHEMATICS AND PROGRAMMING

A Scoping Review of Studies on Computational Thinking in K–12 Mathematics Classrooms

Daniel Hickmott[1] · **Elena Prieto-Rodriguez**[1] · **Kathryn Holmes**[2]

Published online: 12 December 2017

Abstract Since the 1960s, a few, yet very influential, educational researchers have investigated how computer programming can be used to foster mathematics learning. However, since the term 'computational thinking' was popularised by Jeannette Wing in 2006, the number of studies in this area has grown substantially. In this article, we present a systematic analysis of literature linking mathematics education to computational thinking in an attempt to quantify the breadth and depth of existing work in the area. Our analysis indicates that many studies: (1) originate from computer science academics rather than education experts; (2) involve mathematics but mainly concentrate on teaching programming skills; (3) present small-scale research designs on self-reported attitudes or beliefs; (4) rarely deal with concepts in mathematical domain areas such as probability, statistics, measurement or functions. Thus, we conclude that there are opportunities for rigorous research designs reporting on observable learning outcomes, explicitly targeting mathematics, conducted by multidisciplinary teams, and focusing on less-explored domain areas. We believe that these opportunities should be investigated, in order to provide a broader evidence base for developing meaningful digital learning experiences in mathematics for school-aged children.

Keywords Computational thinking · Programming · Mathematics · Digital technologies · Scoping review

Electronic supplementary material The online version of this article (https://doi.org/10.1007/s40751-017-0038-8) contains supplementary material, which is available to authorized users.

✉ Daniel Hickmott
daniel.hickmott@gmail.com

[1] School of Education, University of Newcastle, Callaghan, New South Wales, Australia

[2] School of Education, Western Sydney University, Penrith, New South Wales, Australia

Introduction

Seymour Papert introduced the term *computational thinking* in his 1980 book *Mindstorms: Children, computers, and powerful ideas*. In it, he claimed that students learn most effectively when they are engaged in the construction of something that is meaningful to them and that can be shared. As a mathematician, Papert was particularly interested in the teaching of mathematics and envisioned a learning environment (which he referred to as 'Mathland') that students could use to explore abstract mathematical concepts in a more concrete way. This idea led him to develop the programming language Logo, which he described as, "an instrument designed to help change the way you talk about and think about mathematics and writing and the relationship between them" (Papert 1990, p. 7). Papert believed that students programming in Logo would be able to develop their understanding of learning and thinking through the process of testing and debugging their ideas in code. This vision resonated with some teachers and researchers, who saw Logo and Papert's philosophy of education as an "alternative to the prevailing technocentric and behaviourist notions of computer-aided instruction" (Agalianos et al. 2006, p. 241), notions that were common in the 1980s.

During that time, some teachers were inspired by Papert's *Mindstorms* to make changes to their teaching approaches, and soon Logo programming became part of national educational reforms in both the USA and the UK (Agalianos et al. 2006). Initially, some such changes were at a grassroots level, but in the years following it became part of national educational reforms in both countries. Microcomputers were also introduced into many classrooms in the UK during the 1980s, which Agalianos and colleagues argued was largely motivated by industrialists, rather than in the spirit of true educational reform. The view that computers could be used by students to explore mathematical ideas as Papert envisioned was largely absent. Furthermore, the versions of Logo that were compatible with the most common microcomputer used in classrooms (the BBC Micro) were difficult to install and had limited functionality. Ultimately, Logo was "stripped of its radical potential" (p. 241), and for many teachers Logo became synonymous with 'turtle graphics'. Consequently, in both the UK and the USA, teachers mainly used Logo for tasks that were to be carried out in the classroom, rather than as an environment within which students could develop their thinking.

Almost thirty years after *Mindstorms* was published, Jeanette Wing (2006) wrote an article on computational thinking, which she defined as "thinking like a computer scientist" (p. 35). She argued that computational thinking was a skill that could benefit everyone, not just computer scientists, and that it should be taught alongside "reading, writing and arithmetic" (p. 33). In the decade following her article, the teaching of computer science (CS) in school became more widespread and began to be introduced into compulsory K–12 curricula (Wing 2016). Countries that have recently introduced such curricula, with a focus on computational thinking, include Australia (Falkner et al. 2014), England (Brown et al. 2014), the United States (Fisher 2016) and New Zealand (Bell et al. 2012a).

Programming and computational thinking are generally considered to be separate skills, but programming requires the use of computational thinking and is often used to teach it (Lye and Koh 2014). Programming is the act of writing code that instructs a computer to perform some actions, whereas computational thinking is a "problem-solving methodology" (Barr and Stephenson 2011, p. 48). The teaching of

computational thinking does not necessarily require students to create programs, however: for example, CS Unplugged (Bell et al. 2012b) comprises a set of resources developed to teach this skill without the use of a computer.

Introducing programming and computational thinking into compulsory K–12 classes has presented some challenges for educators, as the learning of these skills has not usually been part of a teacher's formal education. This is particularly true for primary school teachers, as they are unlikely to have completed a technology major and are usually generalist teachers (Vivian et al. 2014). One of the approaches suggested for helping prepare teachers for such teaching is to integrate computational thinking into the subjects they are currently teaching, for example, mathematics (Barr and Stephenson 2011).

Existing Reviews of Computational Thinking Literature

Reviews of the computational thinking literature have been conducted in the past, but none of them have been focused on finding links between computational thinking and the learning of K–12 mathematics. Lye and Koh (2014) searched a pair of educational research databases for studies related to computational thinking in education and, after discarding non-empirical and irrelevant studies, selected twenty-seven for review. They were interested in how computational thinking (through the use of programming) had been incorporated into K–12 curricula, the reported performance outcomes of the studies' participants and the types of interventions that had been used in the reviewed studies. However, this review did not provide any information on how computational thinking had been linked to mathematics learning outcomes. Of the studies they selected for review, nine of these had study participants who were K–12 students and only two of these nine studies involved integration of computational thinking and mathematics. Furthermore, the performance outcomes they reported were related to the computational thinking dimensions defined by Brennan and Resnick (2012) – computational concepts, computational practices and computational perspectives – and not subject content knowledge.

In another pertinent study, Kalelioğlu et al. (2016) searched six databases, four of which contained multidisciplinary research and two CS research, to attempt to find all computational thinking studies published and indexed between 2006 and 2014, and developed a framework from a review of the studies found in the search. In this systematic review, they classified one hundred and twenty-five selected papers according to their "purpose", "targeted population", "emphasised theoretical/conceptual backgrounds", "suggestions of definitions" (of computational thinking), "chosen framework/scope", and the type of paper and "employed research design". Forty-seven had K–12 students as the targeted population and a small amount (2%) of all the papers reviewed included a definition of computational thinking that included mathematical reasoning as an aspect of this term. The links between computational thinking and mathematical reasoning were not considered in this systematic review. They concluded that the computational thinking literature is "at an early stage of maturity" and that the most of the studies they reviewed did not have "research designs" (p. 591), perhaps trying to note a perceived lack of well-designed methods.

Similarly, Falkner et al. (2014) conducted a semi-systematic review of CS education literature, concluding that rigorous research was lacking in its results and found that

most studies took place outside of classrooms. They searched two databases, ACM Digital Library and Google Scholar, for studies published between 2003 and 2013 on Computer Science K–12 education and found a total of seventy-one studies. The classifications of the studies included: the type of research methods used, the context of the study and the presence of concepts from Australia's Digital Technologies curriculum (the equivalent of England's Computing curriculum). Four of the reviewed studies were classified as being conducted in the context of learning mathematics, but it was not clear what the research methods or assessment approaches were in these studies. They also discussed the potential for research into CS education that involved ideas that have been previously studied in mathematics education, such as "gender-based stereotypes and achievement" (p. 9). These reviews all found a dearth of empirical research providing evidence of the transfer effects of programming to the learning of twenty-first- century skills (Scherer 2016). None of the existing reviews of collections of the computational thinking literature have focused on how computational thinking and the learning of mathematics have been linked.

Limitations of this Study

As a team of mathematics educators and computer scientists, we are interested in the links between computational thinking and the learning of mathematics in K–12 education: in this study, we review existing literature to discover how these links have occurred. However, we acknowledge that using the term 'computational thinking', misses important research that was conducted prior to the popularisation of this term. Essential foundational research exploring the use of programming for the teaching of mathematics from the 1980s, expanding on Papert's work with Logo, but conducted prior to 2006, will not be present in this review.

Reports on projects that used computer programming to help children explore mathematics dating back to the 1960s have not been captured in the review presented in this article (for example, Feurzeig et al. 1969; Papert 1972). Also missing from our study is extensive research conducted in the 1980s and 1990s in England and the United States. Most of this research was conducted by proponents of constructionism, a learning theory developed by Papert that built on Piaget's constructivism (Ackermann 2001). For instance, in the United Kingdom, Hoyles and Noss (1992) explored the use of Logo programming and its potential integration into K–12 mathematics education. In the United States, Kafai and Harel (1991) investigated the learning outcomes of students who developed and designed computer games to teach their peers mathematics concepts. Wilensky and Resnick (1999) created StarLogo, a software environment for creating agent-based simulations inspired by Logo, and studied the use of this software for computational modelling in K–12 mathematics and science education. These researchers have continued to work on research projects that do combine these, such as ScratchMaths (Benton et al. 2016; Benton et al. 2017), constructionist gaming (Kafai and Burke 2015) and NetLogo (Tisue and Wilensky 2004).

Outside of the work conducted by these researchers, however, it is unclear how computational thinking and the learning of mathematics are being linked in the wider literature. Some researchers (for example, Barr and Stephenson 2011; Weintrop et al. 2016) have suggested approaches for integrating computational thinking with existing K–12 curricula, including mathematics. The scoping review reported in this article

sheds light on all existing work depicting such an integration since 2006. The aim of our study is to respond to the following questions:

1. What peer-reviewed studies have been published from 2006 to 2016 in relation to computational thinking in K–12 educational contexts?
2. Do these studies link computational thinking to the learning of mathematics and, if so, in which ways?

Methods for this Scoping Review

The approach for this scoping review was designed according to methods discussed by Arksey and O'Malley (2005), who identified four common reasons for conducting a scoping review study. The reasons for conducting this particular scoping review are "to examine the extent, nature and range of research activity" and "to identify research gaps in the existing literature" (pp. 6–7). In this section, the process for searching the databases is explained followed by the methods used for classifying the studies for analysis.

Search Results

Six databases (see Table 1) were searched to find peer-reviewed studies relevant to computational thinking in K–12 education. Four of these databases were multidisciplinary: Springer, ProQuest, ScienceDirect and EBSCO Megafile Premier. These databases included results from education databases, for example ProQuest includes results from the Education Resources Information Center (ERIC) database, as well as STEM and computer science databases. The other two databases searched, IEEE Xplore and ACM Digital Library, focus on computer science and software engineering.

The search term used on each database was "*computational thinking*" *AND* "*school**" *AND (*"*Primary*" *OR* "*Elementary*" *OR* "*Secondary*" *OR* "*High*" *OR* "*K–12*"*)*. The results were limited to peer-reviewed articles published between 2006 and 2016.

After exporting the results into an Endnote library, studies that were irrelevant to computational thinking in K–12 education were excluded and annotated. The steps are

Table 1 Number of results from each selected database

Database	Number of results
Springer	291
IEEE Xplore	283
ProQuest	182
ACM Digital Library	142
ScienceDirect	76
EBSCO Megafile Premier	43
Total	1017

shown in Fig. 1 and can be summarised as follows. Firstly, duplicate studies ($n = 96$), resulting from the search being conducted across different databases, were removed. We also removed studies ($n = 120$) that did not have full text available when the search was conducted. Next, the titles and abstracts of the remaining studies were screened for relevance, and were removed if they did not relate to computational thinking in K–12 education. Following the screening of titles and abstracts, full texts of all remaining studies were examined to remove irrelevant articles ($n = 385$). Studies that were relevant to computational thinking in K–12 education, but that were works-in-progress or that were not written in English, were also removed ($n = 23$). The number of papers removed during these steps, and the reasons that the studies were not relevant or excluded, are shown in Tables 2 and 3.

Following this process, 393 papers were deemed to be relevant to computational thinking in K–12 education. For detailed information on these articles, and the specifications for the selection categories referred to, see the Appendix (Online Resource 1).

This process is summarised in Fig. 1.

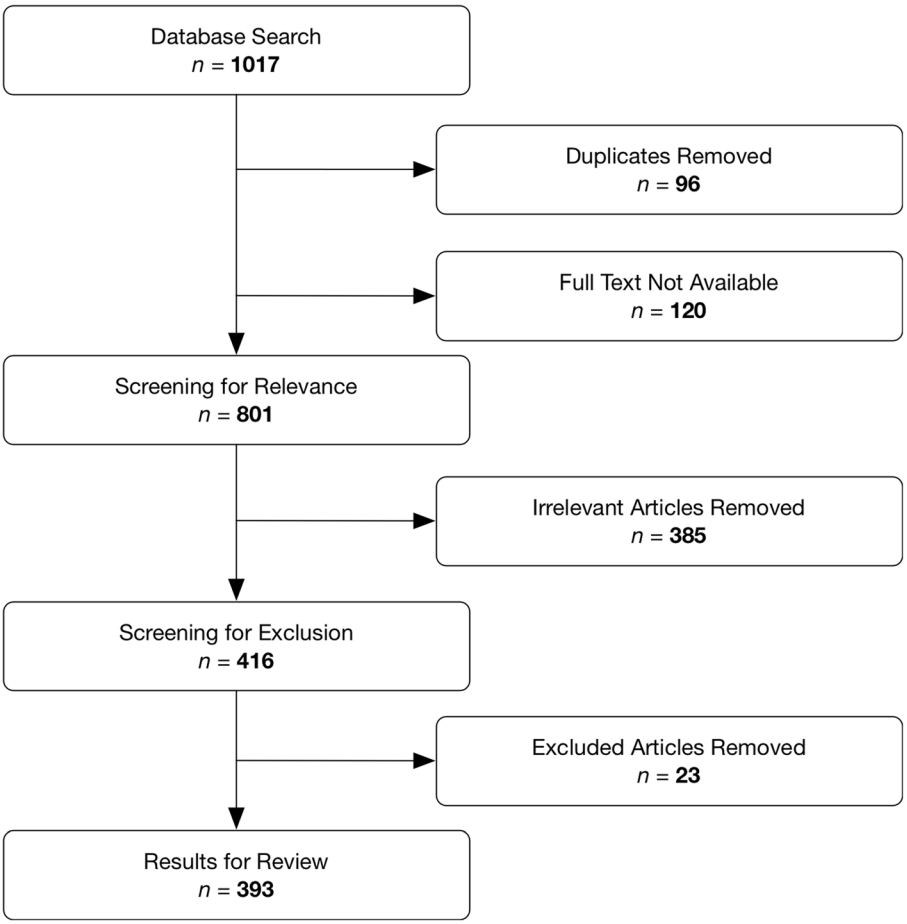

Fig. 1 The review process

Table 2 Number of papers that were marked as not relevant and the reasons for their removal

Reason for study being marked as not relevant	Number of papers
Involved study of undergraduates or other tertiary level	120
Not education related, for example: a paper on a computer science topic with no relation to schools	81
Not a paper, for example: a keynote or preface	76
Involved education research but no real integration of computational thinking	67
Study on a topic completely unrelated to computational thinking and education	27
An evaluation or case study of an e-learning software package	11
A study of pre-school students	3
Total	385

To verify that the study could be reproduced and to ensure validity, two researchers' reviews of the studies were compared. Initially, one of the researchers conducted the review process on the 801 results, after removing duplicate and studies for which there was no full text available. One of the authors reviewed 7% ($n = 60$) of the results and the classifications were compared. After a first round of coding, a level of 78% inter-rater reliability was achieved and it became apparent that some areas of coding needed clarification. This was accomplished and a second iteration of coding was undertaken. As there were multiple measures that were coded for each study, the authors only had to code one field differently for a disagreement to occur. Commonly, this difference was only minor and agreement was reached after briefly discussing the difference. For example, there were a few studies where the authors had not identified the same mathematics domain areas present in the study, but had only disagreed on one of the domain areas. On the second iteration, a 100% inter-coder agreement was achieved. The remaining studies ($n = 393$) were classified according to their *Link to the Learning of Mathematics*, *Activity Approach* and *Mathematics Domain Area*. The four-step approach for classifying the studies is explained in the following subsections.

Step 1: Link to the learning of mathematics

The studies were first classified according to their type of *Link to Learning of Mathematics* as: *Not at All* linked, *Incidentally* linked or *Explicitly* linked (see Fig. 2).

Table 3 Number of papers that were excluded and the reasons for their exclusion

Reason for study being excluded	Number of papers
Not in English	11
Work in progress paper	10
Position paper	2
Total	23

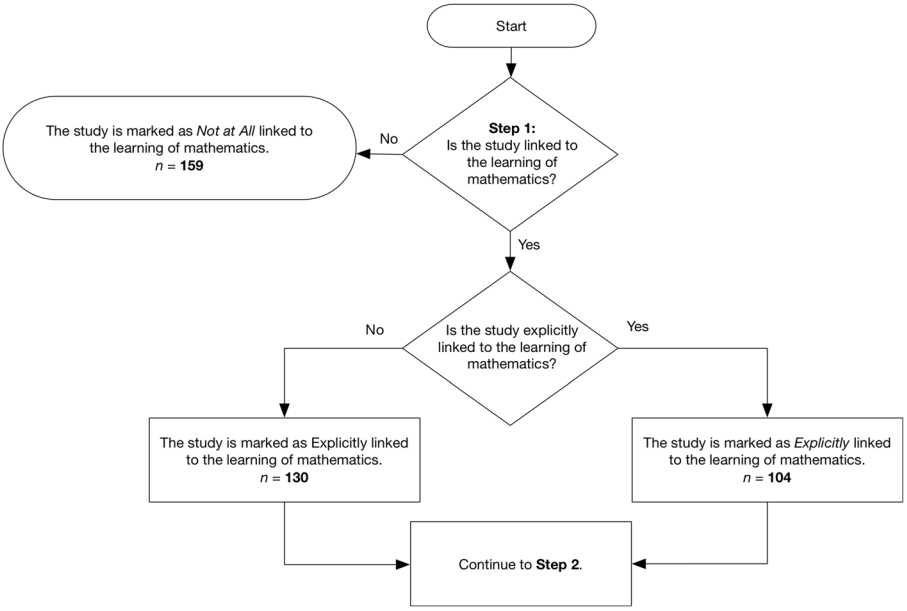

Fig. 2 Step 1 in the classification process

Studies that were incidentally linked to the learning of mathematics ($n = 130$) were those where mathematics concepts were present, but where there was no evidence that the researchers had intended for students to learn these concepts. For instance, some studies involved the design and creation of games, involving visual programming tools such as Scratch that required the use of co-ordinate geometry (Akcaoglu 2014; Pinto and Escudeiro 2014; Repenning et al. 2010), but the researchers did not explicitly set out to teach this concept. Studies that were explicitly linked to the learning of mathematics ($n = 104$) were those in which researchers had made a clear connection between the learning of at least one mathematics concept and computational thinking. For example, Mensing et al. (2013) described several strategies for integrating computational thinking into the US mathematics curriculum. The rest of the studies ($n = 159$) were removed for consideration in the subsequent classification process, as they were found to have no links to the learning of mathematics.

Step 2: Classification of mathematics domain area

Studies that were either *explicitly* or *incidentally* linked to the learning of mathematics were further classified by the *Mathematics Domain Area* present in them. Mathematics concepts were considered present in the studies if there was some evidence of the teaching of, or intention to teach, the concept through the use of computational thinking or programming. The mathematics concepts were grouped into the five following domain areas:

- **numbers and operations** ($n = 196$): counting, operations, number systems and fractions;
- **algebra** ($n = 180$): abstraction of concepts from *Numbers and Operations* and equations;

- **measurement and functions** ($n = 65$): ratios, proportional, linear, non-linear relationships not including trigonometric functions;
- **geometry** ($n = 128$): shapes, Cartesian co-ordinates and area;
- **statistics and probability** ($n = 36$): data, its measurement and representation.

Step 3: Nature of the study

This step, illustrated in Fig. 3, involved checking whether the study was empirical in nature. Studies were thus marked as non-empirical or empirical, and in the latter case we noted the size of the sample and the methods used. In the case of studies dealing explicitly with teaching mathematics, the methods were coded as quantitative, qualitative or mixed.

Step 4: Activity approach

Empirical studies that were *explicitly* linked to the learning of mathematics were classified by the type of knowledge claims that researchers tried to impart when teaching the mathematics concepts: *Procedural*, *Conceptual* or *Both* (where there was evidence of procedural and conceptual knowledge). There were two additional *Activity approach* classifications used for empirical studies explicitly linked to the learning of mathematics. These were *Not Clear,* for studies where there was an empirical component but insufficient information to determine whether the intervention involved *Procedural* or *Conceptual* knowledge. This step in the classification process is depicted in Fig. 4.

It can be difficult to make an exact distinction between *conceptual* and *procedural* knowledge. In this review, the *Activity Approach* has been classified according to the distinctions described by Haapasalo and Kadijevich (2000, p. 141) who defined conceptual knowledge as "knowledge of and a skilful 'drive' along particular networks, the elements of which can be concepts, rules […] and even problems […] given in various representation forms", and added that conceptual knowledge "typically requires conscious thinking". Thus, a student demonstrating conceptual knowledge of a mathematical idea will understand the underlying concepts and relationships between these concepts, and will be able to able to explain why a problem involving this idea can be solved, not just how to solve it. An

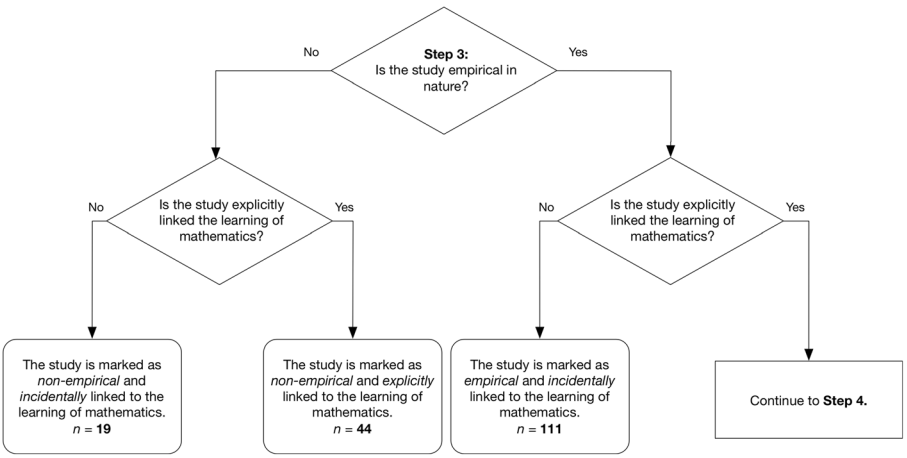

Fig. 3 Step in of the classification process

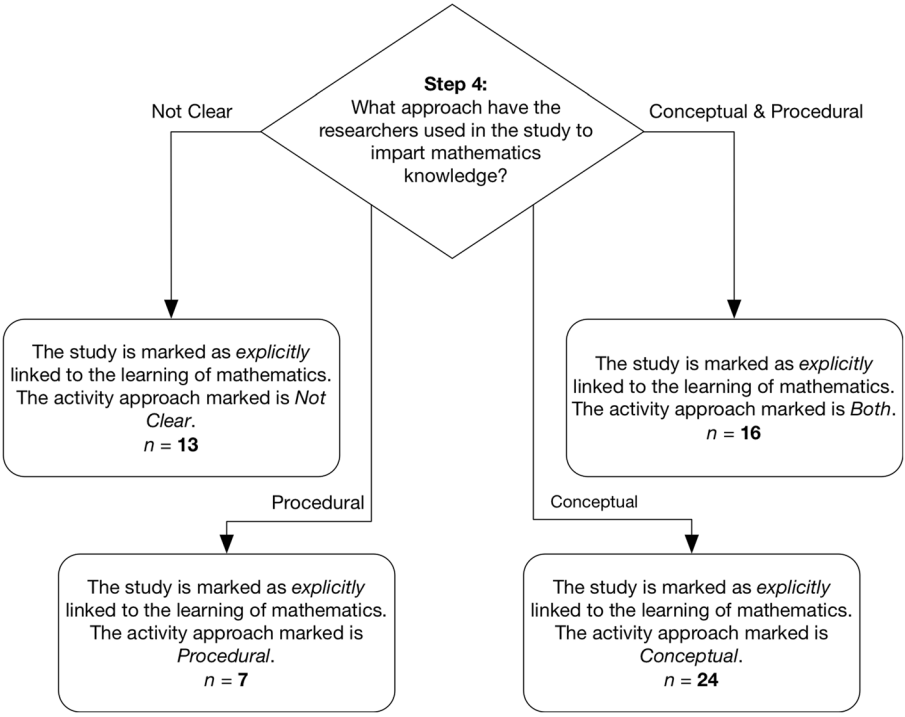

Fig. 4 Step 4 in the classification process

example of study where conceptual knowledge was evident involved an investigation of students' use of ViMAP (a visual programming language) to create agent-based simulations and how this impacted on their conceptual understanding of the relationships among distance, speed and acceleration (Farris and Sengupta 2013).

Haapasalo and Kadijevich define *procedural* knowledge as the "dynamic and successful utilization of particular rules, algorithms or procedures within relevant representation form(s)" (p. 141) and argue that this type of knowledge often "calls for automated and unconscious steps" to be made by students to solve a problem. For example, Xiaoxia and Zhurong (2011) describe a task where high school students calculated the area of a circle by translating the equation into an algorithm written in C program code, but did not include evidence on how students conceptualised the relationships among the equation's variables when writing this program.

Some of the studies reviewed also contained evidence of the use of both conceptual and procedural approaches in their task. Layer et al. (2012) organised a three-week summer camp for high school students that was focused on teaching CS. One of the tasks in this summer camp involved teaching students about how mapping and GPS software is implemented. Students were taught about the "great circle distance formula used to find the shortest distance between two points on the surface of a sphere" (p. 3), and then calculated distances by hand using the formula before implementing the formula in code (an instance of a *procedural* approach). In another task, the students were taught about different encryption algorithms, then devised their own algorithms and paired with other students to test them (an instance of a *conceptual* approach).

Analysis of Relevant Studies

The 393 studies that were determined to be relevant were classified using the process detailed in the previous section. In this section, our findings are presented and discussed. We begin with outlining a broad perspective on the reviewed studies and then focus on the studies that make an explicit link between computational thinking and the learning of mathematics.

Database Analysis

The studies relevant to this review originated from six academic databases. The frequency of the different databases of origin for the 393 relevant studies is shown in Fig. 5. The three databases that contained most of the reviewed studies were: IEEE Xplore, Springer and ACM Digital Library.

Over half (53%) of the reviewed studies originated from the two databases with a CS focus. Consequently, many of the reviewed studies were conducted by CS academics with an interest in K–12 Education. These studies often involved introducing school students and teachers to computational thinking and programming as part of summer camps (e.g. Jimenez and Gardner-McCune 2015; Al-Duwis et al. 2013), teacher workshops (Jiangjiang et al. 2015) and as part of K–12 classes (Nikou and Economides 2014). Additionally, CS academics often ran interventions aimed at improving students' perceptions of CS and to inform them about the potential careers (Al-Duwis et al. 2013; Larkins et al. 2013).

Studies Explicitly Linking Computational Thinking and Mathematics

The links that the researchers made between computational thinking and the learning of mathematics were classified for the 393 relevant studies, according to whether they were *explicitly, incidentally* or *not at all* linked. The percentages of the studies linked to the learning of mathematics are shown in Fig. 6. The majority of the studies (73%) reviewed were not explicitly linked and studies with no link were more common (40%) than those only linked incidentally (33%).

Studies that *incidentally* linked computational thinking to mathematics were more common than those that *explicitly* linked them. Studies were considered to be *incidentally* linked when there was some evidence of mathematics concepts present in the study's intervention or in the authors' discussion. Fundamental programming concepts

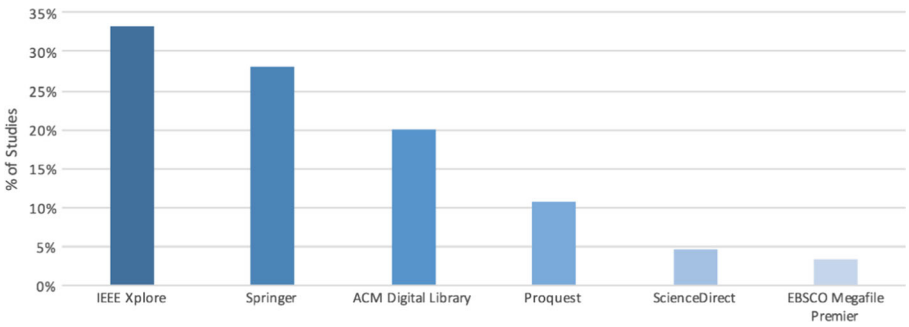

Fig. 5 The databases that relevant results originated from

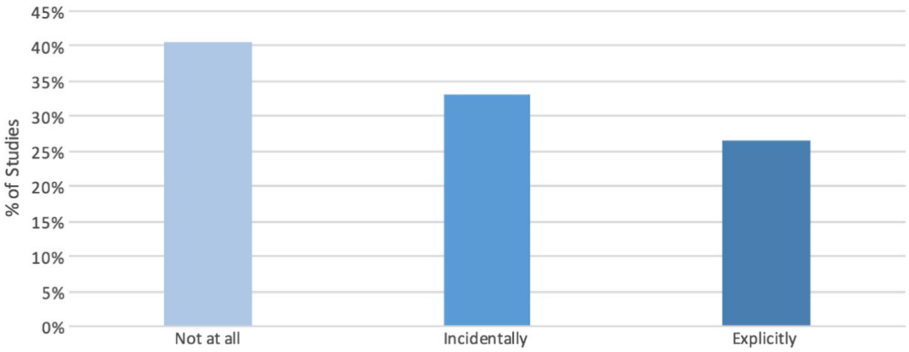

Fig. 6 Degree of linkage in studies between computational thinking and the learning of mathematics

like variables and performing number operations (such as addition and subtraction) were often used in these studies introducing students to computational thinking and programming (Werner et al. 2012; Atmatzidou and Demetriadis 2016; McCoid et al. 2013). Generally, the purpose of the application of these concepts, however, was not to impart mathematical content knowledge to participants, but rather to introduce programming concepts. Thus, there were many studies that did *incidentally* include mathematics concepts, particularly in the domain areas of *Numbers & Operations* and *Algebra*. Generally, the tasks and technologies used to impart mathematics and programming concepts were similar in studies that were linked to the learning of mathematics *incidentally* and *explicitly*. However, in *explicitly* linked studies, there was a difference in the researchers' intent to address mathematics concepts, instead of just programming concepts, or the study reported on learning outcomes in mathematics, rather than learning in regard to computational thinking. For example, Perdikuri (2014) and Al-Humoud et al. (2014) both involved students creating mobile apps (using the AppInventor software) and reported on change in the participants' perceptions of computer science. However, in Al-Humoud et al. (2014), which was explicitly linked to the learning of mathematics, the participating students were instructed to build an educational game app for teaching other students arithmetic.

The reviewed studies did not only involve the use of programming to teach computational thinking and mathematics, however. Unplugged tasks, in which computational thinking concepts were taught without programming a computer, were also present in studies. In the majority of studies that involved them, tasks from the popular resource CS Unplugged were used. Most of these tasks involve some mathematics and have evolved from, and been inspired by, resources focused on learning mathematics (Bell et al. 2012b). For example, the Binary Numbers task involves teaching students about the differences between representing numbers in decimal and binary systems. However, in the *explicitly* linked and *incidentally* linked studies that involved unplugged activities, there were rarely any assessments of how these tasks related to participants' learning of mathematics or computational thinking. Usually, the analysis of these studies was focused on other measures, such as participants' workshop feedback (Al-Duwis et al. 2013; Nishida et al. 2008), and not focused on how these unplugged tasks contributed to participants' learning of mathematics.

In one of the studies that *explicitly* linked mathematics and computational thinking (Dorling and White 2015), the authors describe a language titled *LOGO Unplugged*,

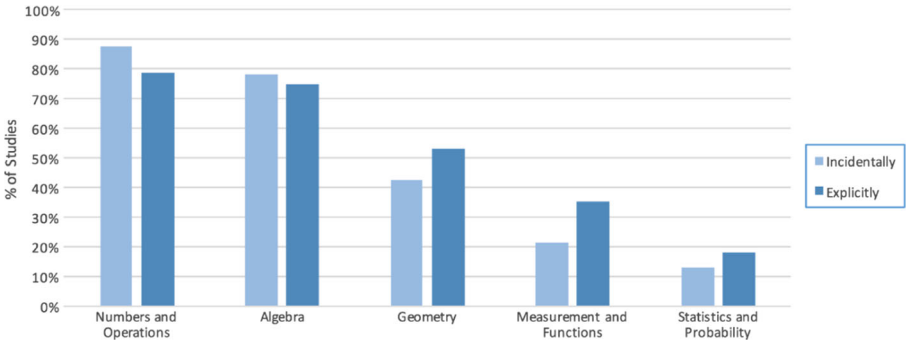

Fig. 7 Domain areas present in studies linked to learning of mathematics

which students can write and compute with using paper and pencil. *LOGO Unplugged* is proposed as a way of teaching unplugged computational thinking and mathematics, with a focus on geometry. However, they only include ancedotal evidence in their study and did not investigate students learning mathematics with this language. In his Masters thesis, which was not captured in this scoping review, Rodriguez (2015) found that CS Unplugged tasks had a positive effect on students' computational thinking. However, he did not investigate the effect of CS Unplugged activites on students' mathematics learning. The absence of studies that *explicitly* link mathematics learning to computational thinking, and that also involve unplugged approaches to computing, suggests this requires further investigation.

Connections to Different Domain Areas of Mathematics

Those studies which were found to be *incidentally* or *explicitly* linked to the learning of mathematics were classified by domain area. The results of this classification are shown in Fig. 7. *Numbers and Operations (*in 88% of *incidentally linked and* 79% of *explicitly linked)* and *Algebra* (in 79% of *incidentally* linked and 75% of *explicitly* linked) were the two most common domain areas present. *Measurement and Functions* (in 22% of *incidentally* linked and 36% of *explicitly* linked) and *Statistics and Probability (*in 13% of *incidentally linked and* 18% of *explicitly linked)* were rarely present in studies linked to the learning of mathematics.

There was often evidence of multiple mathematics domain areas being present in the studies. *Numbers and Operations*, *Algebra* and *Geometry* often appeared together in many of them, particularly where there was an intervention in which students designed games, programmed robots or used turtle geometry environments. Examples of these types of tasks from the reviewed studies included designing a maze game in Scratch (Akcaoglu 2014), programming a robot arm to pick up objects (Kurebayashi et al. 2008) and using turtle geometry to draw patterns for physical fabrication (Turbak et al. 2012). These tasks involved programming to position and move objects around the Cartesian plane, and thus required the application of variables, operations and geometry.

Measurements and Functions and *Statistics and Probability* were the least common domain areas present in the reviewed studies. This could be because *Numbers and Operations* and *Algebra* were often present because they are part of fundamental programming concepts, and because many of the programming environments used in

K–12 involve *Geometry* concepts. *Measurement* was commonly present in studies where participants used and interpreted data collected from sensors – for example, collecting temperature readings from a sensor (Phalke and Lysecky 2010; Brady et al. 2014). *Functions* were often present in studies where students created models, particularly those for physics concepts, such as simulations of Newton's laws of motion (Dukeman et al. 2013; Aiken et al. 2013). *Probability* was common in studies that involved simulations as well. For example, a student calculated "probability to estimate the direction of fire spreading" in a simulation created in AgentSheets (Koh et al. 2013, p. 599), and students modified probabilities used in a model and observed the effect on the simulation (Repenning et al. 2013). *Statistics* were commonly present in studies conducted in the context of science education and usually focused on the use of computational thinking when students interpreted data collected from experiments (Kim 2016; Borne 2010).

Twelve of the reviewed studies had evidence of all five mathematics domain areas present in them. Eight of these studies were explicitly linked to mathematics (8% of all explicitly linked studies) and 4 of these studies were incidentally linked to mathematics (3% of all incidentally linked studies). None of these studies consisted of a single task that involved all five mathematics domain areas. These studies either involved a combination of tasks, which covered different domain areas, that were part of a course or workshop content (Jiangjiang et al. 2015; Chatzinikolakis and Papadakis 2014; Bojic and Arratia 2015), or the studies discussed the integration of computational and programming into mathematics curricula (Sengupta et al. 2013; Reeping and Reid 2015; Dagienė 2008).

Tasks Approaches in Empirical Studies

Studies that were *explicitly* linked to mathematics in Step 3 were subsequently classified by the approach used in the tasks conducted in each study, the *Activity Approach*. The results of this classification are illustrated in Fig. 8. More than a third of the studies were non-empirical (42% of the explicitly linked studies). The authors of these studies would often discuss the relationship between mathematics and computational thinking, as well as high-level mappings of mathematics concepts to tasks, but they did not collect data to support their arguments.

Studies that involved an intervention with a focus on imparting conceptual knowledge were the second most common (23%). The central vision of Papert's work with Logo was

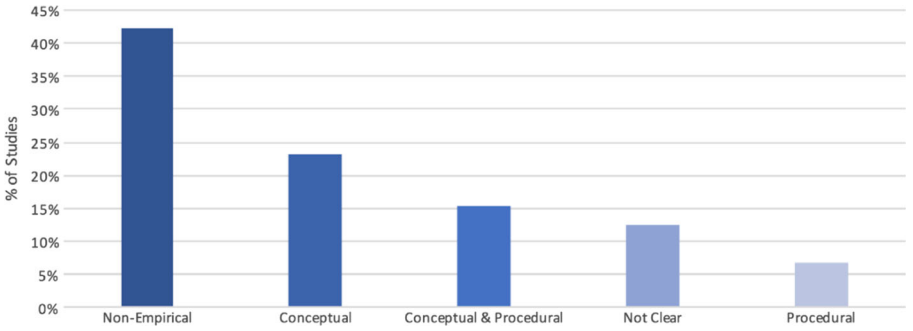

Fig. 8 Activity approaches of studies explicitly linked to Mathematics

to give students another way of reasoning about and conceptualising mathematics (Papert 1996). The legacy of Papert's vision was evident in many of the reviewed studies that were *explicitly* linked, particularly those where the authors incorporated tasks that aimed to increase students' conceptual understanding of *Geometry* through computational thinking and programming (see, for example, Kyriakides et al. 2016).

Studies that were classified with a *Not Clear Activity Approach* had an empirical component, but did not provide enough information about the intervention or tasks to determine the *Activity Approach*. Often these were studies that only briefly outlined the activities presented to teachers and/or students at a workshop and reported on perceptions of CS (Sullivan et al. 2015; Ahamed et al. 2010).

Empirical Studies with Evidence of Impact on Students' Outcomes

The empirical studies that were *explicitly* linked to the learning of mathematics were reviewed to identify which had included evidence of participants' learning outcomes in mathematics ($n = 60$). Only studies where researchers had used quantitative methods were considered for this last part of the analysis. It is important to emphasise here that we consider non-quantitative studies essential to the development of the field, but we found that they all dealt with small samples and therefore would not be considered representative by some government agencies in certain countries in terms of broad educational reform. The research methods used in each of these studies are shown in Table 4.

The 41 studies that contained quantitative analysis were examined to identify the type of evidence that was gathered, the results of which are shown in Fig. 9. The most common type of evidence gathered in the studies was students' perceptions of CS and related careers. Six of the studies were focused on an evaluation of a tool, course or workshop. For example, Ruutmann (2014) surveyed Estonian secondary school STEM in order to evaluate teacher education courses that incorporated mathematics, programming and computational thinking. Self-reported learning outcomes and attitudes were also reported in three of the studies: for example, Ke (2014) found positive changes in students' dispositions towards mathematics as a result of creating games.

There were only eleven studies that reported participants' learning outcomes in mathematics. None of these studies involved the observation of long-term learning outcomes and also conducted an experiment with randomisation of participants. Four of these studies contained only descriptive statistics, which were used to report on students' results in assessments. Six of them employed both descriptive and inferential statistics, two of which were focused on investigating the correlation between students' scores on computational thinking assessments and mathematics test scores (Oliveira et al. 2014; Lewis and Shah 2012).

Table 4 Research methods used in empirical studies explicitly linked to the learning of mathematics

Research Methods	Number of Studies
Quantitative Only	24
Qualitative Only	19
Mixed Methods	17
Total	60

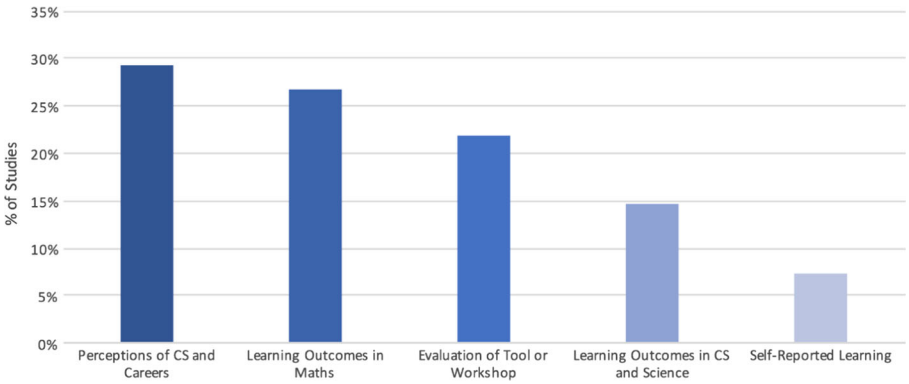

Fig. 9 Type of evidence gathered in quantitative studies

Of these eleven studies, five involved an intervention, contained both descriptive and inferential statistics, and reported on a change in learning outcomes in mathematics. One of these studies reported on a change in one group of students' understanding of fractal geometry by comparing results from pre-test and post-test questionnaires (Wilkerson-Jerde 2014). Another study compared students' performance on pre-test and post-test programming quizzes by gender (Kalelioğlu 2015). Only three of these five studies had a research design with control and experimental groups: Babbitt et al. (2015), Boyce et al. (2011) and Calao et al. (2015).

Babbitt et al. (2015) compared the performance of Ghanaian students in two groups on pre-tests and post-tests that measured students' understanding of geometric concepts. The focus of this study was on investigating whether students who learned concepts using a Culturally Situated Design Tool (CSDT) performed better on the tests than those who learned concepts using an existing mathematics instructional software package. The CSDT that was used in the study was a modified version of the *Snap!* visual programming environment, which was modified to allow the students to write code that drew Ghanaian Adinkra symbols. The study reported results from an evaluation that took place in a Ghanaian junior high school with a total of nineteen students. The students were split into a control group with ten students, who received instruction with the GeoGebra software package, and an intervention group of nine students, who received instruction with CSDT. The intervention group performed significantly higher in the post-test than those in the control group, which supports the argument made by Babbitt and colleagues that a culturally relevant approach to instruction can increase students' performance. As the sample size in the study was small and the focus of the study was on the CSDT, rather than the link between computational thinking and mathematics learning, it is difficult to make wider claims about how these can be combined in practice from this study's findings.

The second quasi-experimental study, Boyce et al. (2011), compared the impact of a game and an on-line tool on students' understanding of co-ordinate geometry. The game, titled BeadLoom Game (BLG), was a gamified version of the on-line tool, titled Virtual BeadLoom (VBL). In BLG and VBL, students could use iteration to create virtual bead art drawn on a Cartesian plane. The study reported results from two experiments conducted at summer camps held in the USA. The first of these experiments involved 21 middle school students as participants and the second 22 high school

students. The experiments were conducted using a "switching replications experimental design" (p. 245): the participants were split into two groups and received both BLG and VBL interventions. The students were given a pre-test and post-test, as well as a test in the middle of the experimental period, before the groups switched between using BLG and VBL. Boyce and colleagues were mainly concerned with a comparison of the effectiveness of the game and the tool, and concluded that adding gaming elements to the tool resulted in a significant increase in students' motivation and enjoyment, as well as their understanding of co-ordinate geometry. The focus of the study was largely on students' motivation and enjoyment rather than on the effect of incorporating computational thinking into the teaching of mathematics. The results from this study do indicate, however, that mathematical concepts can be combined with computing ones (iteration) and have a positive effect on students' learning outcomes and motivation.

The third quasi-experimental study, Calao et al. (2015), examined the effect of learning to code in Scratch on students' understanding of mathematical processes. This study was conducted using a quasi-experimental design, and employed a pre-test and a post-test to measure students' learning invoking four different mathematical processes: "*modelling*", "*reasoning*", "*problem solving*" and "*exercising*" (p. 20). There were 42 participants in the study, all sixth-grade (11–12 years old) Colombian students, who were divided into a control group ($n = 18$) and an experimental group ($n = 24$). The experimental group was given an intervention for three months providing students with an introduction to programming concepts in Scratch and the opportunity for them to develop their own games and simulations. There was no change in the approaches used to teach mathematics to the control group, which the authors refer to as the "traditional" approach (p. 24).

Calao and colleagues found that the post-test scores of students in the experimental group were significantly higher than those in the control group, indicating that learning to code in Scratch was more effective for teaching mathematical processes than the 'traditional' approach. There is a potential caveat in this study, however. The content of the post-test assessment and its relation to the material delivered to the experimental group are not explained in the study. One explanation for the high scores in the experimental group could be that the post-test had questions directly related to the skills students learned in the intervention, but Calao et al. (2015) do not elaborate this in their discussion. Despite this caveat, the results of the study do provide some evidence that students learning computational thinking through coding in Scratch can have positive effects on their ability to use mathematical processes.

These findings indicate that there is a lack of quasi-experimental research that *explicitly* links the learning of mathematics (as evidenced by student academic outcomes) with computational thinking. We believe that this is an area that needs to be investigated to build an evidence base for combining mathematics and computational thinking in practice.

Conclusion

The introduction of computational thinking skills into K–12 curricula presents many challenges and opportunities for educators committed to improving students' understanding of mathematics. The analysis of existing literature presented in this article signals areas of research that could potentially be broadened in order to provide teachers, outreach providers and professional development agencies an evidence base

for developing digital learning experiences in mathematics that create for children a "range of opportunities to engage as a bricoleur or bricoleuse in activities with scientific and mathematical content" (Papert 1993, p. 145).

In summary, there were four main gaps identified in this review of the computational thinking literature published between 2006 and 2016. These identified gaps are a lack of:

- multidisciplinary research projects that involve both education and computer science researchers;
- research into approaches to teaching concepts in the domain areas of probability, statistics, measurement and functions in conjunction with computational thinking;
- empirical studies that include concrete ideas or practices for K–12 educators that *explicitly* link the learning of mathematics and computational thinking;
- studies that link computational thinking and mathematics explicitly and involve analysis of students' long-term learning outcomes in mathematics, for sample sizes larger than fifty.

Many of the studies were found in specialised computer science data bases, conducted by researchers in the computer science field as part of outreach or summer camps that involved teaching students computational thinking, usually through the introduction of programming languages. These studies were often centred on improving students' perceptions of CS and related careers or evaluating feedback from participants, rather than the intellectual effect of the intervention on students' learning. There appears to be a dearth of research originating from academics with an educational research background. To us, this suggests that there is an opportunity for more multidisciplinary research to be conducted, informed both by computer science and by educational research literature.

Studies that *explicitly* linked the learning of mathematics concepts to computational thinking were uncommon in the reviewed literature. There was often evidence of concepts involving numbers, operations or algebra being engaged with in the studies reviewed, but this was usually with the intention of introducing programming concepts. This seems to suggest (something which is confirmed by prior work conducted by the likes of Papert, Noss, Hoyles, Harel, Wilensky and Resnick) that there are opportunities to investigate explicit ways with which to enhance the understanding of mathematics concepts using the computational thinking. In particular, more research needs to be conducted into ways of using computational thinking for the teaching concepts in the domain areas of probability, statistics, measurement and functions.

Non-empirical studies that *explicitly* linked the learning of mathematics to computational thinking were common in the reviewed studies. These studies often contained discussion of potential mappings between these two areas, but most did not include concrete ideas or practices that could be applied by K–12 educators in domain areas other than geometry. This is another literature void that could be addressed by future research in the field.

Empirical studies that *explicitly* link the learning of mathematics to computational thinking and whose researchers reported on students' learning outcomes in mathematics were rare in the reviewed literature. Those studies that did report on students' learning outcomes in mathematics were often short-term, one-off studies with no long-term learning outcomes. The authors of this scoping review are aware that an independently

evaluated quasi-empirical study, ScratchMaths, is currently underway in the UK, the first of its kind (Benton et al. 2017). Studies *explicitly* targeting the learning of mathematics, such as ScratchMaths, with a focus on domain areas less explored to date, and conducted by multidisciplinary teams, need to occur in other parts of the world if we are to provide empirical evidence of the transfer effects of programming to the learning of mathematics. The lack of studies in this area points to the conceptual and methodological difficulties involved in designing multi-disciplinary research aiming to examine student learning outcomes across a number of separate, yet linked, domains. Our review found that there were more studies that *incidentally* link mathematics learning with computational thinking than those where the link with computational thinking was made explicit. This could indicate that there is a lack of mathematics education expertise in the research teams conducting many of these studies. Therefore, to advance our knowledge of effective ways of delivering integrated curricula in schools across the mathematics and computer science domains, it is essential for research teams to draw on expertise which can bridge discipline silos without diminishing the importance of any constituent part.

References

Ackermann, E. (2001). Piaget's constructivism, Papert's constructionism: What's the difference. *Future of Learning. Group, 5*(3), 438–449.

Agalianos, A., Whitty, G., & Noss, R. (2006). The social shaping of Logo. *Social Studies of Science, 36*(2), 241–267.

Ahamed, S., Brylow, D., Ge, R., Madiraju, P., Merrill, S., Struble, C., & Early, J. (2010). Computational thinking for the sciences: A three-day workshop for high school science teachers. In *Proceedings of the 41st ACM technical symposium on computer science education* (pp. 42–46). Milwaukee: SIGSCE.

Aiken, J., Caballero, M., Douglas, S., Burk, J., Scanlon, E., Thoms, B. & Schatz, M. (2013). Understanding student computational thinking with computational modeling. In Engelhardt, P. Chunkin, A., & Rebello, S. (eds.), *AIP Conference Proceedings* (vol. 1513 issue 1, pp. 46–49). Melville: AIP Publishing.

Akcaoglu, M. (2014). Learning problem solving through making games at the game design and learning summer program. *Educational Technology, Research and. Development, 62*(5), 583–600.

Al-Duwis, M., Al-Khalifa, H., Al-Razgan, M., Al-Rajebah, N., & Al-Subaihin, A. (2013). Increasing high school girls' awareness of computer science through summer camp. Paper presented at the 2013 Global Engineering Education Conference. Berlin: IEEE.

Al-Humoud, S., Al-Khalifa, H., Al-Razgan, M., & Alfaries, A. (2014). Using App Inventor and LEGO mindstorm NXT in a summer camp to attract high school girls to computing fields. Paper presented at the 2014 Global Engineering Education Conference. Istanbul: IEEE.

Arksey, H., & O'Malley, L. (2005). Scoping studies: Towards a methodological framework. *International Journal of Social Research Methodology, 8*(1), 19–32.

Atmatzidou, S., & Demetriadis, S. (2016). Advancing students' computational thinking skills through educational robotics: A study on age- and gender-relevant differences. *Robotics and Autonomous Systems, 75*(Part B), 661–670.

Babbitt, W., Lachney, M., Bulley, E., & Eglash, R. (2015). Adinkra mathematics: A study of ethnocomputing in Ghana. *REMIE Multidisciplinary. Journal of Educational Research, 5*(2), 110–135.

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.

Bell, T., Newton, H., Andreae, P., & Robins, A. (2012a). The introduction of computer science to NZ high schools: An analysis of student work. In *Proceedings of the 7th Workshop on Primary and Secondary Comupting education* (pp. 5–15). Hamburg: WIPSCE.

Bell, T., Rosamond, F., & Casey, N. (2012b). Computer science unplugged and related projects in math and computer science popularization. In H. Bodlaender, R. Downey, F. Fomin, & D. Marx (Eds.), *The multivariate algorithmic revolution and beyond* (pp. 398–456). Berlin: Springer-Verlag Berlin Heidelberg.

Benton, L., Hoyles, C., Kalas, I., & Noss, R. (2016). Building mathematical knowledge with programming: Insights from the ScratchMaths project. In A. Sipitakiat & N. Tutiyaphunegprasert (Eds.), *Proceedings of constructionism 2016* (pp. 25–32). Bangkok: Suksapattana Foundation.

Benton, L., Hoyles, C., Kalas, I., & Noss, R. (2017). Bridging primary programming and mathematics: Some findings of design research in England. *Digital Experiences in Mathematics Education, 3*(2), 115–138.

Bojic, I. & Arratia, J. (2015). Teaching K–12 students STEM-C-related topics through playing and conducting research. In Proceedings of the 2015 Frontiers in Education Conference (pp. 548–555). El Paso: IEEE.

Borne, K. (2010). Astroinformatics: Data-oriented astronomy research and education. *Earth Science Informatics, 3*(1), 5–17.

Boyce, A., Campbell, A., Pickford, S., Culler, D., & Barnes, T. (2011). Experimental evaluation of BeadLoom game: How adding game elements to an educational tool improves motivation and learning. In *In Proceedings of the 16th annual joint conference on innovation and technology in computer science education*. Darmstadt: ACM.

Brady, C., Holbert, N., Soylu, F., Novak, M., & Wilensky, U. (2014). Sandboxes for model-based inquiry. *Journal of Science Education and Technology, 24*(2), 265–286.

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. In *Paper presented at the 2012 annual meeting of the American Educational Research Association*. Vancouver: AERA.

Brown, N., Sentance, S., Crick, T., & Humphreys, S. (2014). Restart: The resurgence of computer science in UK schools. *ACM Transactions on Computing Education, 14*(2), 1–22.

Calao, L., Moreno-León, J., Correa, H., & Robles, G. (2015). Developing mathematical thinking with scratch: An experiment with sixth-grade students. In G. Conole, T. Klobučar, C. Rensing, J. Konert, & E. Lavoué (Eds.), *Design for teaching and learning in a networked world* (pp. 17–27). Cham: Springer.

Chatzinikolakis, G. & Papadakis, S. (2014). Motivating K–12 students learning fundamental computer science concepts with App Inventor. *Paper presented at the 2014* International conference on interactive mobile communication technologies and learning. Thessaloniki: IMCL.

Dagienė, V. (2008). Teaching information technology and elements of informatics in lower secondary schools: Curricula, didactic provision and implementation. In R. Mittermeir & M. Sysło (Eds), Informatics education: Supporting computational thinking (pp. 293–304). Berlin: Springer-Verlag Berlin Heidelberg.

Dorling, M., & White, D. (2015). Scratch: A way to Logo and python. In *Proceedings of the 46th ACM technical tymposium on computer science education* (pp. 195–196). Kansas City: SIGCSE.

Dukeman, A., Caglar, F., Shekhar, S., Kinnebrew, J., Biswas, G., Fisher, D. & Gokhale, A. (2013). Teaching computational thinking skills in C_3STEM with traffic simulation. In A. Holzinger & G. Pasi (Eds), Human–computer interaction and knowledge discovery in complex, unstructured, big data (pp. 350–357). Berlin: Springer-Verlag Berlin Heidelberg.

Falkner, K., Vivian, R., & Falkner, N. (2014). The Australian digital technologies curriculum: Challenge and opportunity. In J. Whalley & D. D'Souza (Eds.), *Proceedings of the Sixteenth Australasian Computing Education conference* (pp. 3–12). Auckland: Australian Computer Society.

Farris, A. & Sengupta, P. (2013). On the aesthetics of children's computational modeling for learning science. In *Proceedings of the 12th International Conference on Interaction Design and Children* (pp. 479–482). New York. ACM.

Feurzeig, W., Papert, S., Bloom, M., Grant, R. & Solomon, C. (1969). Programming-languages as a conceptual framework for teaching mathematics. (Final report on the first fifteen months of the LOGO project.) Washington, DC. (http://nla.gov.au/nla.cat-vn5207614).

Fisher, L. (2016). A decade of ACM efforts contribute to computer science for all. *Communications of the ACM, 59*(4), 25–27.

Haapasalo, L., & Kadijevich, D. (2000). Two types of mathematical knowledge and their relation. *Journal für Mathematik-Didaktik, 21*(2), 139–157.

Hoyles, C., & Noss, R. (1992). *Learning mathematics and Logo*. Cambridge: MIT Press.

Jiangjiang, L., Wilson, J., Hemmenway, D., Yingbo, X., & Cheng-Hsien, L. (2015). Oh SNAP! A one-week summer computing workshop for K–12 teachers. In *In Paper presented at the 10th international conference on Computer Science and Education*. Cambridge: IEEE.

Jimenez, Y., & Gardner-McCune, C. (2015). Using app inventor and history as a gateway to engage African American students in computer science. In *In Paper presented at the Research in Equity and Sustained Participation in Engineering, Computing, and Technology (RESPECT)*. Atlanta: IEEE.

Kafai, Y., & Burke, Q. (2015). Constructionist gaming: Understanding the benefits of making games for learning. *Educational Psychologist, 50*(4), 313–334.

Kafai, Y., & Harel, I. (1991). Children's learning through consulting: When mathematical ideas, programming knowledge, instructional design, and playful discourse are intertwined. In I. Harel & S. Papert (Eds.), *Constructionism* (pp. 85–100). Norwood: Ablex.

Kalelioğlu, F. (2015). A new way of teaching programming skills to K–12 students: Code.org. *Computers in Human Behavior, 52*, 200–210.

Kalelioğlu, F., Gülbahar, Y., & Kukul, V. (2016). A framework for computational thinking based on a systematic research review. *Baltic Journal of Modern Computing, 4*(3), 583–596.

Ke, F. (2014). An implementation of design-based learning through creating educational computer games: A case study on mathematics learning during design and computing. *Computers & Education, 73*, 26–39.

Kim, H. (2016). Inquiry-based science and technology enrichment program for middle-school-aged female students. *Journal of Science Education and Technology, 25*(2), 174–186.

Koh, K., Repenning, A., Nickerson, H., Endo, Y., & Motter, P. (2013). Will it stick? Exploring the sustainability of computational thinking education through game design. In *Proceedings of the 44th ACM technical symposium on computer science education* (pp. 597–602). Denver: SIGCSE.

Kurebayashi, S., Aoki, H., Kamada, T., Kanemune, S. & Kuno, Y. (2008). Proposal for teaching manufacturing and control programming using autonomous mobile robots with an arm. In R. Mittermeir & M. Sysło (Eds), Informatics education: Supporting computational thinking (pp. 75–86). Berlin: Springer-Verlag Berlin Heidelberg.

Kyriakides, A., Meletiou-Mavrotheris, M., & Prodromou, T. (2016). Mobile technologies in the service of students' learning of mathematics: The example of game application a.L.E.X. In the context of a primary school in Cyprus. *Mathematics Education Research Journal, 28*(1), 53–78.

Larkins, D., Moore, J., Rubbo, L., & Covington, L. (2013). Application of the cognitive apprenticeship framework to a middle school robotics camp. In *Proceedings of the 44th ACM technical symposium on computer science education* (pp. 89–94). Denver: SIGCSE.

Layer, R., Sherriff, M. & Tychonievich, L. (2012). "Inform, experience, implement": Teaching an intensive high school summer course. In *Paper presented at the 2012* Frontiers in Education Conference. Washington, DC: IEEE.

Lewis, C., & Shah, N. (2012). Building upon and enriching grade four mathematics standards with programming curriculum. In *Proceedings of the 43rd ACM technical symposium on computer science education* (pp. 57–62). Raleigh: SIGCSE.

Lye, S., & Koh, J. (2014). Review on teaching and learning of computational thinking through programming: What is next for K–12? *Computers in Human Behavior, 41*, 51–61.

McCoid, S., Freeman, J., Magerko, B., Michaud, C., Jenkins, T., McKlin, T., & Kan, H. (2013). EarSketch: An integrated approach to teaching introductory computer music. *Organised Sound, 18*(2), 146–160.

Mensing, K., Mak, J., Bird, M. & Billings, J. (2013). Computational, model thinking and computer coding for US common Core standards with 6- to 12-year-old students. In Proceedings of the Emerging eLearning Technologies and Applications Conference (pp. 17–22). Stary Smokovec: IEEE.

Nikou, S. & Economides, A. (2014). Transition in student motivation during a scratch and an app inventor course. In Proceedings of the 2014 Global Engineering Education Conference (pp. 1042–1045). Istanbul: IEEE.

Nishida, T., Idosaka, Y., Hofuku, Y., Kanemune, S., & Kuno, Y. (2008). New methodology of information education with "computer science unplugged". In R. Mittermeir & M. Sysło (Eds.), *Informatics education: Supporting computational thinking* (pp. 241–252). Berlin: Springer-Verlag Berlin Heidelberg.

Oliveira, O., Nicoletti, M., & Cura, L. (2014). Quantitative correlation between ability to compute and student performance in a primary school. In *Proceedings of the 45th ACM technical symposium on computer science education* (pp. 505–510). Atlanta: SIGCSE.

Papert, S. (1972). Teaching children to be mathematicians versus teaching about mathematics. *International Journal of Mathematical Education in Science and Technology, 3*(3), 249–262.

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

Papert, S. (1990). *A critique of technocentrism in thinking about the school of the future*. Cambridge: Epistemology and Learning Group, MIT Media Laboratory.

Papert, S. (1993). *The children's machine: Rethinking school in the age of the computer*. New York: Basic Books.

Papert, S. (1996). An exploration in the space of mathematics educations. *International Journal of Computers for Mathematical Learning, 1*(1), 95–123.

Perdikuri, K. (2014). Students' experiences from the use of MIT app inventor in classroom. In *Proceedings of the 18th Panhellenic conference on informatics*. Athens: ACM.

Phalke, A., & Lysecky, S. (2010). Adapting the eblock platform for middle school STEM projects: Initial platform usability testing. *IEEE Transactions on Learning Technologies, 3*(2), 152–164.

Pinto, A., & Escudeiro, P. (2014). The use of scratch for the development of 21st-century learning skills in ICT. In *In Paper presented at the 9th Iberian Conference on Information Systems and Technologies*. Barcelona: IEEE.

Reeping, D. & Reid, K. (2015). Viewing K–12 mathematics and science standards through the lens of the first-year introduction to engineering course classification scheme. In Paper presented at the 2015 Frontiers in Education Conference. Washington, DC: IEEE.

Repenning, A., Webb, D., & Ioannidou, A. (2010). Scalable game design and the development of a checklist for getting computational thinking into public schools. In *Proceedings of the 41st ACM technical symposium on computer science education* (pp. 265–269). Milwaukee: SIGCSE.

Repenning, A., Basawapatna, A. & Klymkowsky, M. (2013). Making educational games that work in the classroom: A new approach for integrating STEM simulations. In *Paper presented at the Games Innovation Conference*. Vancouver: IEEE.

Rodriguez, B. (2015). *Assessing computational thinking in Computer Science Unplugged activities*. Unpublished Masters thesis. Golden: Colorado School of Mines.

Ruutmann, T. (2014). Optional STEM courses for secondary schools designed and implemented for enhancement of K–12 technology education, in order to excite students' interest in technology and engineering education. In *Paper presented at the 2014* International Conference on Interactive Collaborative Learning (ICL). Dubai.

Scherer, R. (2016). Learning from the past: The need for empirical evidence on the transfer effects of computer programming skills. *Frontiers in Psychology, 7*, 1390.

Sengupta, P., Kinnebrew, J., Basu, S., Biswas, G., & Clark, D. (2013). Integrating computational thinking with K–12 science education using agent-based computation: A theoretical framework. *Education and Information Technologies, 18*(2), 351–380.

Sullivan, K., Byrne, J., Bresnihan, N., O'Sullivan, K. & Tangney, B. (2015). CodePlus: Designing an after-school computing programme for girls. In *Paper presented at the 2015* Frontiers in Education Conference. Washington, DC: IEEE.

Tisue, S. & Wilensky, U. (2004). NetLogo: A simple environment for modeling complexity. In Minai, A. & Bar-Yam, Y. (eds.), *Proceedings of the 5th international conference on complex systems* (pp. 16–21). Boston.

Turbak, F., Sandu, S., Kotsopoulos, O., Erdman, E., Davis, E., & Chadha, K. (2012). Blocks languages for creating tangible artifacts. In *Proceedings of the 2012 symposium on visual languages and human-centric computing* (pp. 137–144). Innsbrucka: IEEE.

Vivian, R., Falkner, K., & Falkner, N. (2014). Addressing the challenges of a new digital technologies curriculum: MOOCs as a scalable solution for teacher professional development. *Research in Learning Technology, 22*. https://doi.org/10.3402/rlt.v22.24691.

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.

Werner, L., Denner, J., Campe, S., & Kawamoto, D. (2012). The fairy performance assessment: Measuring computational thinking in middle school. In *Proceedings of the 43rd ACM technical symposium on computer science education* (pp. 215–220). Raleigh: SIGCSE.

Wilensky, U., & Resnick, M. (1999). Thinking in levels: A dynamic systems approach to making sense of the world. *Journal of Science Education and Technology, 8*(1), 3–19.

Wilkerson-Jerde, M. (2014). Construction, categorization, and consensus: Student-generated computational artifacts as a context for disciplinary reflection. *Educational Technology Research & Development, 62*(1), 99–121.

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

Wing, J. (2016). Computational thinking, 10 years later. www.microsoft.com/en-us/research/blog/computational-thinking-10-years-later/.

Xiaoxia, W. & Zhurong, Z. (2011). The research of situational teaching mode of programming in high school with Scratch. In *Proceedings of the 2011 Conference on Information Technology and Artificial Intelligence* (pp. 488–492). IEEE.