Computational Thinking and Mathematics

Laura Broley, Chantal Buteau, Simon Modeste, Maryna Rafalska, Max Stephens

Handbook of digital (curriculum) resources in mathematics education
Section: How digital resources transform content
Chapter: Computational thinking and mathematics
Authors (in alphabetical order):
Laura Broley, Chantal Buteau, Simon Modeste, Maryna Rafalska, & Max Stephens
Contact: simon.modeste@umontpellier.fr

Abstract: This chapter deals with the recent development of Computational Thinking (CT) in the curricula of many countries, principally in mathematics. It aims at discussing, broadly, how CT changes the mathematical activity and impacts mathematical contents. More precisely, we study the relations between mathematics, computer science, mathematical thinking and computational thinking and discuss new content at the interface of computer science and mathematics as well as the transformation of mathematical content due to the integration of CT. We ground our discussions on examples of integration of CT and computer science in various countries, we further discuss the origins of CT in mathematics education, and offer an epistemological reflection on the disciplines of mathematics and computer science. Finally we discuss related issues for classroom implementation, teaching resources, and teacher development.

Keywords: Computational thinking; mathematics; curriculum; resources; teacher education

1. Introduction

The rapid development of digital technologies has created both the opportunity and the need to transform the content students learn in schools and universities. Proponents of computational thinking (CT)[1] argue that curricular content should include elements of Computer Science (CS) that would enable students to not only understand and critically use digital technologies but also create them to solve problems in their everyday lives. Students thus need to study the underpinnings of digital technologies (e.g., the algorithms that make them work) and to learn how to create and use them to solve problems (e.g., through computer programming).

Let us start by clarifying the necessity of discussing CT in this handbook concerning *digital (curriculum) resources* in *mathematics education*. Computer programs, programming languages, and programming tools are now common resources for mathematics students and teachers. We consider them to be digital resources in the sense that they involve or stem from the use of digital (computer) technologies and, more generally, have links with CS. Although the delimitation of these digital resources requires some discussion, they have influenced the curricula of many countries and have changed the way we can create, use, and understand mathematics. Since the 1980s, researchers have explored these curricular issues through the lens of CT, which is present explicitly or implicitly in recent curricular evolutions related to CS and digital technologies in mathematics education.

The current section of the handbook asks how digital resources transform content. In activities that engage students in CT and mathematics, this content transformation is often visible, e.g. when programming is involved; however, it is needed to deeply understand the nature of such transformation from various points of view. To begin, we introduce three examples that illustrate some of the possible changes brought about by the introduction of CT in (mathematics) curricula and highlight the diversity of situations and forms in which CT may appear.

1 We return to the meaning of this term in Section 2.

1.1. Example 1: Using parameters and functions to create shapes and patterns in Scratch

Grade 9 students (aged 14–15) in France work through the textbook activity depicted in Figure 1. Students are expected to have worked with Scratch programming since Grade 6 or 7. In Part 1 of the activity, they use Scratch to first define a block of code (called “Square”) that will plot a square with side length and colour as given parameters. For instance, the block “Square” with inputs “100” and “25” should plot a square with side length equal to 100 steps and of colour 25 (which is green in Scratch). In Part 2, students then use the block of code they created to write scripts that will trace out specific shapes and patterns (depicted in Parts 2a, 2b, and 2c at the bottom of Figure 1).

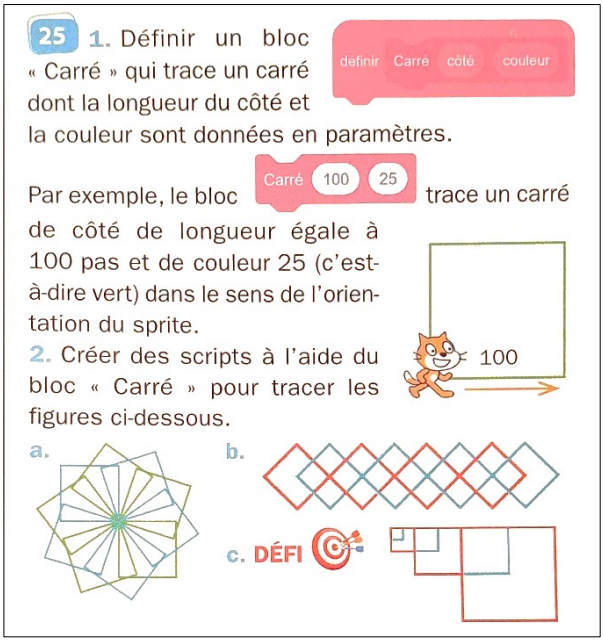

Fig. 1 Activity in a Grade 9 mathematics textbook in France (Brisac et al., 2021, p. 23).

1.2. Example 2: Enacting and thinking about the bubble sort algorithm

A Grade 6 class (students aged 11–12) in Australia is invited to enact the bubble sort algorithm for sorting 10 random numbers (between 1 and 100) from smallest to largest, like in the video snapshot depicted in Figure 2. Ten students stand in a line, each holding their own randomly selected number, in a scrambled order. Another student (a “sorting student”) comes forward, starting at the left of the line and, using pairwise comparisons (addressing each pair sequentially), directs students to stay or to exchange places with the one next to them. After the student has moved along the line of 10 numbers, another student comes forward and repeats the procedure, which continues until all the numbers are sorted from smallest to largest. The class repeats the algorithm for different sets of 10 numbers, keeping a record of the number of sorting students needed. The teacher then provokes an investigation, starting with the question: If 10 students are lined up in a scrambled order, how many students are needed to sort the line of numbers to ensure (i.e., to guarantee) that all 10 students with their numbers are successfully sorted smallest to largest?

Fig. 2 A snapshot from the video of students enacting bubble sort (CS4ALLUSA, 2012).

1.3. Example 3: Using programming to build tools for exploring conjectures

In a first-year university mathematics course in Ontario (Canada), students (aged 18+) are asked to construct a computer program (e.g., using VB.NET or Python) to investigate a conjecture that they have selected or produced about hailstone sequences[2]. The instructor encourages students to work in groups to formulate conjectures and to search online for open conjectures. The code for calculating the length of a hailstone sequence is constructed in class (Figure 3). Then, each student builds on the given code to program interactive environments to investigate their conjectures and write a report on their findings. For example, one student conjectured that there is a relationship between the number of even numbers in a hailstone sequence and the length of the sequence (in particular, the number of even numbers will decrease as the length increases). Figure 4 shows the interface of the program the student created to support their exploration, which led to findings (depicted in their graph) that might encourage a step back from the computer to consider issues related to mathematical proof.

```
length = 0
seed = n 'User inputs n
Do While seed > 1 'Hailstone Sequence
    If seed Mod 2 = 0 Then
        seed = seed / 2
    Else
        seed = seed * 3 + 1
    End If
    length = length + 1
Loop
TextBox2.Text = length
```

Fig. 3 Code (in VB.NET) calculating the length of a hailstone sequence starting at n

2 Start with a positive integer n > 1. Generate the sequence by the following rule: If n is even, divide by 2; if n is odd, multiply by 3 and add 1. If n > 1, repeat the process. If n = 1, stop. For example: 10, 5, 16, 8, 4, 2, 1; this hailstone sequence has length 6 since it has 6 terms after 10.

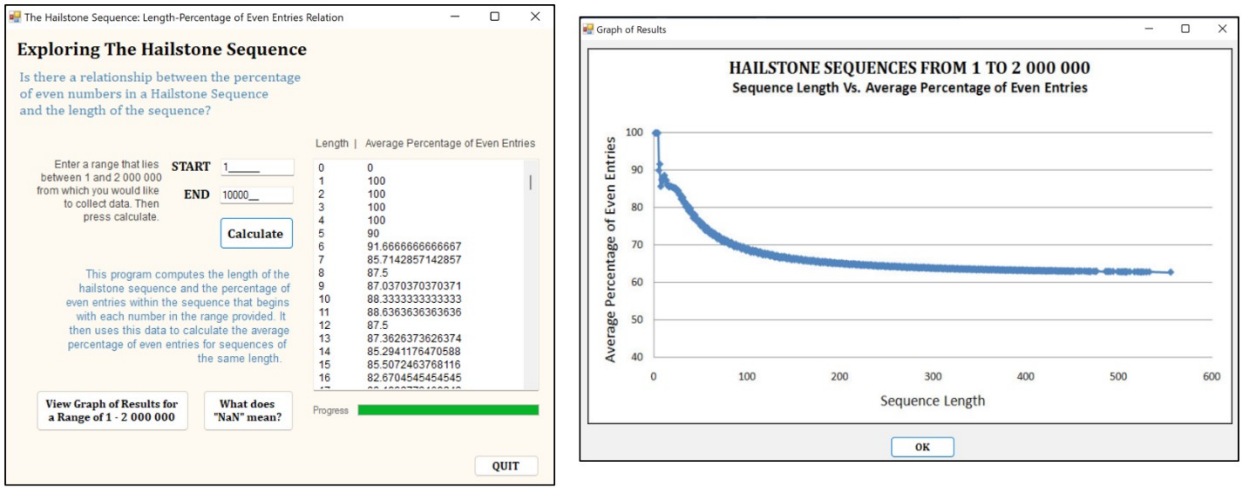

Fig. 4 The interface of a computer program (left) and a graph (right) created by a student to investigate their conjecture about hailstone sequences.

Although these three examples of CT and mathematics are very different, they all illustrate how the development of CS and the omnipresence of digital technologies have transformed mathematics teaching and learning. For instance, parameters, functions, shapes, and patterns are already typically part of mathematics curricula: Example 1 leads to questions about how these concepts are transformed in a computational context and the place of programming in learning mathematics. The study of algorithms now also has an important place in mathematics classrooms: Example 2 points to the new kinds of algorithms and related mathematical questions and concepts that can be explored by mathematics students and teachers, as well as the different ways in which they can be explored (e.g., in an "unplugged" manner, without the use of digital technologies). Similarly, Example 3 highlights how the kinds of mathematics problems and the ways in which students can solve them have been transformed by the potential offered by computer programming.

In this chapter, we draw on these examples to reflect on how the integration of CT in education may transform mathematics content[3] and to discuss related key questions elicited by such transformation. In Section 2, we address the important question of our evolving understanding and relevance of CT in and for (mathematics) education, and how resulting mathematics content transformation has been realized in some jurisdictions. In Section 3, we complement this discussion on CT in mathematics education by exploring the relations between mathematics and CS, as disciplines, in order to enlighten further perspectives to transformed content. In Section 4, we consider pertinent issues related to a successful implementation of CT in mathematics classrooms and a diffusion of mathematics content transformation in teaching practices. Finally, in Section 5 we discuss the key perspectives resulting from the chapter.

2. Evolution of the meaning of CT in and for (mathematics) education

There is yet to be a clear consensus in the literature that has sought to clarify the evolution of the meaning of CT over the years and the different ways in which CT has been conceptualized in international curricula. In this section, we describe its evolution and exemplify the breadth of these conceptions. We do not propose or select a single definition of CT but rather to explore

3 Here, we consider content in a broad view, including notions and concepts, techniques and skills, ways of thinking, and even cultural and general knowledge about the discipline.

possible meanings and highlight similarities, differences, and issues that need to be addressed, including the ways CT may transform and generate mathematics content.

2.1. Evolution of CT amongst scholars

The invention of the personal computer in the 1970s provided opportunities to transform education. Seymour Papert's (1972, 1980) pioneering work developing and implementing the Logo programming language provided a vision of educational opportunities—namely, that students could use programming to turn the computer into an "object-to-think-with" (see e.g. Chapter XX (Francis et al.) of this handbook) and to engage in creative projects that are meaningful to them. According to Weintrop et al. (2016), it was in this context that Papert "was the first to use the term computational thinking to refer to the affordances of computational representations for expressing powerful ideas" (ibid, p. 130), as later elaborated by Papert and Harel (1991). The term CT initially was not linked to CS education, as CS had not yet developed as a school subject. For Papert, CT was a process intertwined with programming and constituted more than being able to use computers; as Barba (2016) explains,

> It's a source of power to *do* something and figure things out, in a dance between the computer and our thoughts. The inversion, starting with computing as a formal thing to understand and then come to the application later, takes away its power. (p. 25)

Far from advocating for a direct teaching of specific programming skills, Papert emphasized the importance of social and affective features of children's programming experiences allowing them to use CT to engage in creative projects across disciplines. Until recently, much of Papert's vision remained as such, possibly due to the then unwelcoming programming environments and teachers' limited access to technology in classrooms (Benton et al., 2017). A shift in education research throughout the 1990s and 2000s focused more on the use of already programmed digital tools, with only sporadic research on CT (not explicitly termed as such) in the mathematics education research literature (Healy & Kynigos, 2010; Lagrange & Rogalski, 2017).

Coming from the discipline of CS, Jeannette Wing (2006) more recently spurred a revival of CT, advocating it as a fundamental 21st century skill (like reading, writing, and arithmetic) that all students should learn. On a basic level, Wing (2006) described CT as thinking like a computer scientist, including (but not limited to) processes such as data representation and abstraction, problem decomposition and reduction, algorithmic and recursive thinking, automation, and simulation. A clear and often cited definition of CT was provided later: "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 [such as a computer]" (as cited in Wing, 2010, p. 1). Since then, many scholars in CS education have elaborated definitions of CT (e.g., Brennan & Resnick, 2012; Curzon et al., 2019; Grover & Pea, 2013; Lee, 2016) and produced several syntheses (e.g., Kallia et al., 2021; Shute et al., 2017), as illustrated in Figure 5.

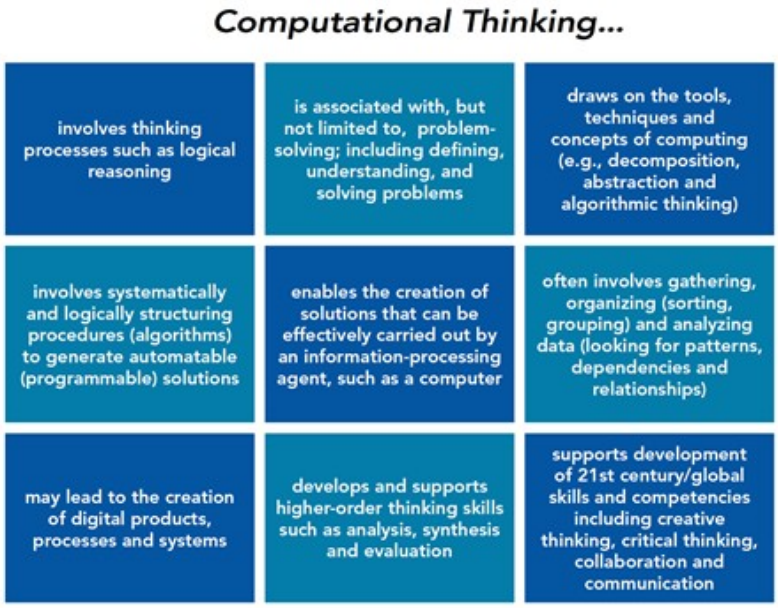

Fig. 5 A synthesis of CT based on research (Let's Talk Science, 2018, p. 5).

diSessa (2018) has criticized computer scientists' and CS educators' lack of attention to the original work by Papert (and others in mathematics education), as well as their representation of CT as a general transferable skill similar to problem solving, with no specific connection to the discipline of mathematics. Kallia et al.'s (2021) recent attempt to characterize CT in mathematics education—involving a systematic literature review (of 56 papers) and Delphi-study (seeking consensus among 25 experts)—identified three key aspects of CT to be addressed in mathematics education:

1. *problem solving* as a fundamental goal of mathematics education in which CT is embedded;
2. *thinking (cognitive) processes* that include (but are not limited to) abstraction, decomposition, pattern recognition, algorithmic thinking, modelling, logical and analytical thinking, generalization and evaluation of solutions and strategies; and
3. *phrasing the solution* of a mathematical problem in such a way that it can be transferred or outsourced to another person or a machine (*transposition*). (p. 179; emphasis added)

In line with diSessa's (2018) critique, these conceptions include aspects (problem solving, abstraction, generalization) that already are considered aims of mathematics education, regardless of the existence of digital resources like computer programs, programming languages, and digital programming tools. In particular, while the presence of digital resources is only anticipated in the third aspect proposed by Kallia et al. (2021), this conception of CT does not specify how that presence may transform mathematical thinking processes or the kinds of problems that can be solved in mathematics.[4]

Other work in mathematics education has focused on how CT arises in mathematical activity and transforms what is possible to do and learn in mathematics. This reflects the transformation that occurred within the discipline of mathematics, as acknowledged by the European Mathematical Society in 2011: "Together with theory and experimentation, a third pillar of scientific inquiry of

4 This conception also does not identify additional elements of CT that may arise from the discipline of CS and that can (or cannot) be addressed in mathematics (education). In Section 3 we explore some such elements that could be incorporated in mathematics classrooms.

complex systems has emerged in the form of a combination of modeling, simulation, optimization and visualisation" (p. 2).

Some studies have more closely examined the place of CT in professional mathematicians' work (e.g., Broley, 2015; Broley et al., 2017; Broley et al., 2018; Lockwood et al., 2016; Lockwood et al., 2019; Modeste, 2012a; Modeste & Ouvrier-Buffet, 2011), including identifying specific practices in which CT may be involved (see Figure 6). The involvement of CT in such practices leads to a transformation of the mathematics that can be done and learned due to the affordances of programmable tools (Gadanidis, 2017); for instance, the automation of mathematical processes that allows for dynamic modelling, visualization, and experimentation in interaction with and leading to the refinement of one's own mathematical thinking (Wilensky, 1995). Other examples are the "agency" and "audience" affordances (Gadanidis, 2017); indeed, by articulating relationships between the mathematical and computational concepts involved, students not only create "new" mathematics but also reveal this act of creation to others (Noss & Hoyles, 1996).

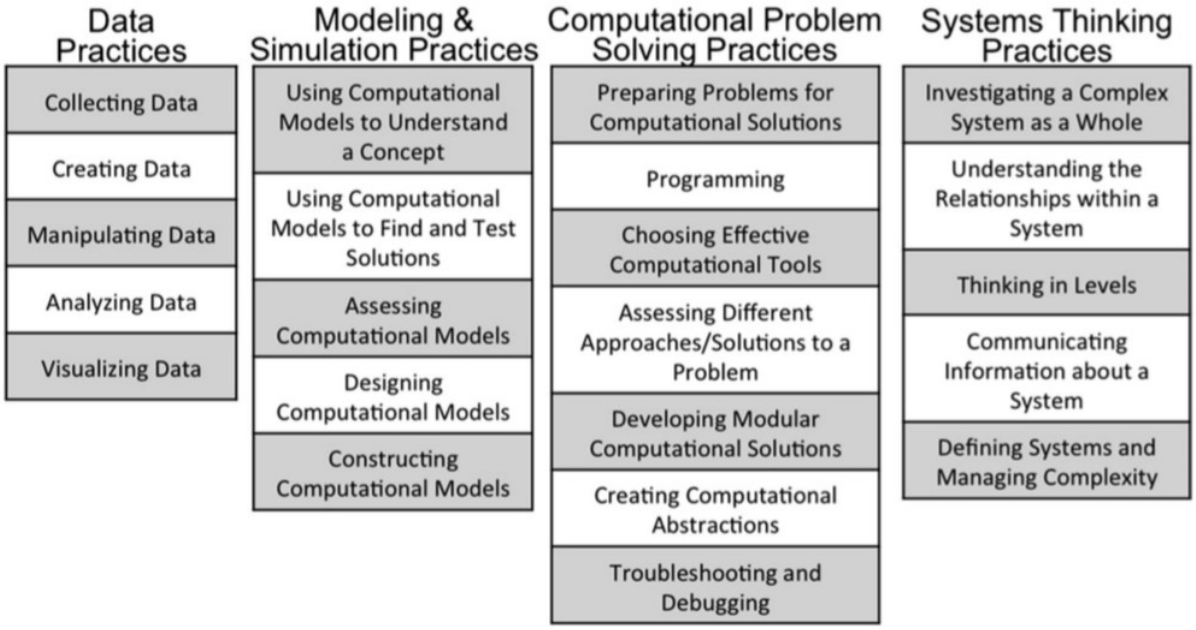

Fig. 6 Taxonomy of CT practices used by mathematicians (Weintrop et al., 2016, p. 135).

We return to Example 3 from Section 1 to illustrate the above conceptualizations of CT.

In this example, a university mathematics student designed a computer program to test a self-formulated conjecture about hailstone sequences: namely, that the percentage of even entries in a hailstone sequence decreases as the length of the sequence increases. To do this, the student had to think up an approach for testing their conjecture and represent the approach in such a way that it could be carried out by a computer. The student decided that their program needed to complete three main tasks: (1) for a given range of initial values, calculate the sequence lengths and the percentage of even entries in each sequence; (2) record the information and use it to calculate the average percentage of even entries for sequences of the same length; and (3) output a table listing the average percentage of even entries for each sequence length. The student then broke down each of these tasks further and represented them in VB.NET code. For instance, to accomplish (1), the student started with the code in Figure 3 and made alterations such as embedding the code within a For Loop over an inputted range of values, adding a count of even entries, and

using the count and the length to calculate the desired percentage. Once the program was complete, the student used it to test their conjecture. This included making the range of initial values as large as possible and studying the tabular output. The student decided that a graphical representation would be helpful and used Excel to produce one (shown in Figure 4). Finally, the student wrote a report of their findings.

Broadly speaking, the student in this example used programming to create an object-to-think-with, thus engaging in CT in Papert's (1980) sense of the word. Aligning with the first key aspect of CT identified by Kallia et al. (2021), the student's CT was embedded in solving a problem: in this case, developing and testing a conjecture. When the student was thinking about how to outsource the testing to a computer, they were engaging in CT in the sense of Wing (2010) or the transposition aspect identified by Kallia et al. (2021). We see how this activity can require the student to engage in cognitive processes that are highlighted by both Wing (2006) and Kallia et al. (2021): e.g., decomposition, algorithmic thinking, and analytical thinking. The activity also engages the student in CT practices shared by professional mathematicians: e.g., most of the computational problem-solving practices from Figure 6, as found by Weintrop et al. (2016). When using their program, the student engaged in various data practices from Figure 6 as well, including creating, manipulating, analyzing, and visualizing data. In the end, the student experienced several affordances of programming for mathematics as described by Gadanidis (2017). For example, the automation of mathematical processes made it possible for the student to produce the desired output for a range of initial values from 1 to 2,000,000. The student exercised agency, from formulating a conjecture that interested them to determining which computational representations they would use to explore it. The programming environment the student created and accompanying report also made the student's work easily shareable with an audience: e.g., the instructors tasked with evaluating their work and the 20–30 peers who explored their own conjectures in weekly 2-hour lab sessions (see Buteau et al., 2022, for an analysis of the teaching–learning environment in which the student worked).

2.2. Evolution of CT in prescribed curricula

Papert believed his vision bypassed what schools did and he was skeptical about what school systems could do. However, one could argue that the evolution of digital technologies (to be more user-friendly and omnipresent), alongside the widespread adoption of the term CT, made it possible to bring Papert's ideas into a form that could be taken over by curriculum writers and shifted into broader educational agendas. As alluded to in Section 2.1, many give credit to Wing for leading the widespread adoption of the term CT, and she was explicit about her goal of inspiring school curricular reform. At that time, in the mid 2000's, there was not yet a concrete roadmap for how it could be done. Later work aimed to address this gap. For instance, the taxonomy proposed by Weintrop et al. (2016; Figure 6) sought to pinpoint CT practices used by professional mathematicians and scientists that could be embedded into specific school curricular areas (in that case, mathematics and science). Another approach has been to show how CT practices identified in the discipline of CS could be mapped onto existing curricular areas. For example, Rich et al. (2019) showed how the CT practices from the U.S. College Board Computer Science Principles could be mapped onto the Common Core State Standards for Mathematics.

The OECD's recent decision to include CT as a fundamental component of PISA's 2021 Mathematics Framework ensures that CT will now be considered in measuring educational progress and achievement in relation to mathematical literacy around the world. CT thus is no

longer seen as an option that schools and national governments can choose to ignore in their curricula:

> Students should possess and be able to demonstrate [CT] skills as they apply to mathematics as part of their problem-solving practice. ... The framework anticipates a reflection by participating countries on the role of [CT] in mathematics curricula and pedagogy. (OECD, 2018, p. 5)

The framework is also likely to direct the way CT will be introduced in many curricula, assuming participating countries will align their curricula with PISA's expectations. The framework mentions several ways of conceptualizing CT in mathematics, based on the literature cited in Section 2.1. It further stipulates: "The nature of [CT] within mathematics is conceptualised as defining and elaborating mathematical knowledge that can be expressed by programming, allowing students to dynamically model mathematical concepts and relationships" (OECD, 2018, pp. 11-12).

In the past decade, curricular reforms in many countries have included CT in compulsory education, starting as early as Grade 1 or age 6 (Bocconi et al., 2016; Dagienė et al., 2019; So et al., 2020; Webb et al., 2017). The elementary curricula in different countries currently integrate CT in different ways. For instance, in England, CT is integrated as part of a distinct Computing curriculum (Benton et al., 2017), while Finland and Sweden use a cross-curricular approach in which CT is expected to be applied in all subjects, while also being taught through an explicit integration of programming in Technology or Crafts and Mathematics (Bocconi et al., 2018). At the secondary level, some countries have a long tradition of teaching CT (without necessarily using this term) in compulsory as well as elective CS courses (e.g., in Ukraine CS has been taught at the secondary level since 1985) but there also is a recent shift towards compulsory integration of CT at that level, either in CS (e.g., in Poland; Webb et al., 2017) or in specific subjects like mathematics (see, for example, the Ontario, Australia, and France case studies below).

At the tertiary level, where curricula are less standardized and not necessarily regulated by governments, the impact of the CT movement is less clear. CS quickly became an academic discipline for those who wish to enter computer-related careers or become software developers. To respond to the various CT needs of students aiming for different careers (e.g., in science, engineering, art, or business), various practitioners have developed specific courses with specific kinds of computing tasks (Guzdial, 2019). In terms of mathematics curricula, a 2016 survey of 46 mathematics departments in the U.K. indicated that a large portion (89%) of undergraduate programs teach CT to all students through mandatory programming components (Sangwin & O'Toole, 2017). Such components have been integrated in different ways—for instance, as separate CS course requirements, within specific computationally driven courses such as numerical analysis or modelling, or as a transversal problem-solving tool across existing courses such as calculus or algebra (Buteau et al., 2022).

There are several reasons to integrate CT in the mathematics curriculum. In addition to the institutional argument related to PISA's new Mathematics Framework, there are more pragmatic arguments, including the advantages of embedding CT in mathematics to avoid slotting new CS courses into already overcrowded curricula and finding qualified teachers to teach them (Hickmott et al., 2018). There also are social arguments, such as reaching more students and broadening participation in computational fields from underrepresented groups (Weintrop et al., 2016). More fundamentally, there is an epistemological argument based on the symbiotic relationship between mathematics and CS, whereby one naturally enriches and essentially

transforms the other (Benton et al., 2017; diSessa, 2018; Papert, 1980; Weintrop et al., 2016). The question remains as to how CT can be integrated in mathematics curricula. Below, we provide three case studies of how this question has been answered in the jurisdictions of Ontario (Canada), Australia, and France.

2.2.1. The case of Ontario (Canada)

Ontario's Grades 1 to 9 Mathematics curricula recently have been revised to include the development of CT through the explicit expectation that students will learn and use coding as part of the algebra strand (Government of Ontario, 2020, 2021a). CT is presented as part of the transferable skills that "are in high demand in today's globally connected world, with its unprecedented advancements in technology" (Government of Ontario, 2021b, p. 30). The Government of Ontario's (2021b) definition of CT resembles that of Wing (2010): "The thought process involved in expressing problems in such a way that their solutions can be reached using computational steps and algorithms" (p. 513).

The overall expectation is that students in Grades 1 to 8 ("elementary school"; ages 5–14) will learn to solve problems and create computational representations of mathematical situations using coding concepts and skills. Specific coding concepts and skills are suggested to be learnt in a progression; for example, sequential events in Grade 1, nested events in Grade 4, and sub-programs and other control structures in Grade 7. By the end of Grade 9 (first year of "secondary school"; ages 14–15), students are expected to be able to apply coding skills to represent mathematical concepts and relationships dynamically, and to solve problems, in algebra and across the other strands (e.g., number, data, spatial sense, and financial literacy). More specifically, Grade 9 students are expected to learn to:

- use coding to demonstrate an understanding of algebraic concepts including variables, parameters, equations, and inequalities;
- create code by decomposing situations into computational steps in order to represent mathematical concepts and relationships, and to solve problems; and
- read code to predict its outcome, and alter code to adjust constraints, parameters, and outcomes to represent a similar or new mathematical situation. (Government of Ontario, 2021a, C2. Coding)

The goal of solving problems aligns with Aspect 1 of the three key aspects of CT to be addressed in mathematics education identified by Kallia et al. (2021); the notion of transposition (Aspect 3) is captured by the expectations that students will read, alter, and create code to represent mathematical situations, concepts, and relationships, but the various cognitive processes mentioned as Aspect 2 are not as evident. In fact, the term CT itself is present only in the front matter or glossary of curricular documents; in the specific expectations for student learning, the focus is on the development of specific coding concepts and skills to be used for mathematical purposes.

At this time, the Grades 10–12 Mathematics curricula, revised in 2005 and 2007, have yet to be revised to align with the introduction of coding in Grades 1–9 Mathematics. Moreover, specific studies of Computer Technology, Computer Programming, and CS are offered through optional secondary school courses (in Grades 9–12), which were revised in 2009 or earlier.

2.2.2. The case of Australia

Another contemporary example of a curricular integration of CT is from Australia, in its nationally approved Mathematics curriculum fromKindergarten to Year 10 (ages 5–16; Australian Curriculum Assessment and Reporting Authority [ACARA], 2022). CT is presented as one of several key processes in mathematics teaching and learning, much like the conceptions of CT offered by Wing (2006) and Kallia et al. (2021; Aspects 1-3).

In its preface, Mathematics: About the Learning Area, the curriculum outlines how:

> Students develop [CT] through the application of its various components: decomposition, abstraction, pattern recognition, use of models and simulations, algorithms and generalisation. [CT] approaches involve experimental and logical analysis, empirical reasoning, and computer-based simulations. The simulations can then be used to generate and test hypotheses and conjectures, identify patterns and key features (or counterexamples), and dynamically explore variation in the behaviour of structures, systems and scenarios. (ACARA, 2022, p. 10)

This introduction to the curriculum then links [CT] to algorithms, software, and the use of digital tools:

> The capacity to purposefully select and effectively use the functionality of a digital device, platform, software or digital resource is a key aspect of [CT] in the Mathematics curriculum. … Different digital tools or platforms can carry out computations and implement algorithms using numerical, textual, statistical, probabilistic, financial, measurement, geometrical, graphical, logical and symbolic functionalities. (ACARA, 2022, p. 10)

In the ensuing content descriptions covering six content areas—number, algebra, measurement, space, statistics, and probability—the term CT is not used explicitly. The content descriptions illustrate the affordances of CT for mathematics learning and mathematical practices (Gadanidis, 2017; Weintrop et al., 2016). These are evidenced by the employment of terms such as *digital tools* (mentioned 46 times), *algorithms* (mentioned 16 times), and *software* (mentioned 10 times) in specific content descriptions. The following examples illustrate such content descriptions:

- In Year 3 (ages 8–9) in statistics, students: create and compare different graphical representations of data sets including using software where appropriate; interpret the data in terms of the context.
- In Year 5 (ages 10–11) in number, students: create and use algorithms involving a sequence of steps and decisions and digital tools to experiment with factors, multiples and divisibility; identify, interpret and describe emerging patterns.
- In Year 7 (ages 12–13) in probability, students: conduct repeated chance experiments and run simulations with a large number of trials using digital tools; compare predictions about outcomes with observed results, explaining the differences.
- In Year 9 (ages 14–15) in space, students: design, test and refine algorithms involving a sequence of steps and decisions based on geometric constructions and theorems; discuss and evaluate refinements.
- In Year 10 (ages 15–16) in algebra, students: experiment with functions and relations using digital tools, making and testing conjectures and generalizing emerging patterns.

These content descriptions are intended to express sequences of learning in Mathematics that complements the Digital Technologies curriculum (Kindergarten to Year 10), showing how the two curricular areas work together to develop students' digital literacy and CT skills. Explicit

links to CS are not the focus of the curriculum in the compulsory years of schooling. These connections emerge more clearly in senior high school (Years 11 and 12) in courses such as Algorithmics, Computer Science, and Applied Computing.

2.2.3. The case of France

In 2009, in France, "algorithmics"[5] became part of the core of the Mathematics curriculum in high school (Grades 10–12, ages 15–18), with the expectation of being taught as transversally applicable across mathematical themes (analysis, algebra, geometry, statistics). Some specific algorithms or algorithmic activities, in connection with specific mathematics topics, were mentioned in the curriculum; for example, the bisection method for finding a root of an equation (in Grade 10), generating simulations for random experiments (in Grades 10–12), or finding thresholds for converging sequences (in Grade 12). This introduction of algorithmics was linked to the general idea of making solutions that can be carried out effectively by a machine through the use of a programming language (e.g., CT in the sense of Wing, 2010). In 2012, a CS option was also added to the scientific paths of high school (in Grade 12).

In the 2016 reform (Ministère de l'Éducation Nationale, de l'Enseignement Supérieur et de la Recherche [MEN], 2015), the curriculum for elementary and middle school was rebuilt on a common core of five domains (Gueudet et al., 2017), to which each discipline (including Mathematics) is expected to contribute. For example, in Cycle 4 (Grades 7–9; ages 12–15), one of the skills to be developed in the first domain (languages for thinking and communicating) is "understanding and expressing using mathematical, scientific, and informatics languages" (MEN, 2015, p. 220, our translation); in the second domain (methods and tools for learning), "the teaching of CS, given in the courses of mathematics and technology, permits to deepen the use of digital tools and learn to progress by trial-and-error" (MEN, 2015, p. 221).

In the revised curricula for Cycle 4 (Grades 7–9; ages 12–15) and the last year of Cycle 3 (Grade 6; age 12), content related to CS can be found in both Mathematics and Technology[6], the teaching of which may involve teachers of both subjects. The accompanying curriculum documents for teachers (MEN, 2016) highlight the importance of developing students' "understanding that computer languages are used for programming digital tools and for automatic treatments of data" and "knowledge of the basic principles of algorithmics and software design" (MEN, 2016, p. 1, our translation).

In the Mathematics curriculum for Cycle 4 (MEN, 2015, pp. 366–380), one of the five themes is "algorithmics and programming," which has a general end-of-cycle goal of enabling students to write, elaborate, and execute a simple program. The "knowledge and skills" associated with this goal include:

- decomposing a problem into sub-problems in order to structure a program, recognizing patterns;
- writing, elaborating (testing, correcting), and executing a program as an answer to a given problem; and
- notions of algorithm and program, variable (in CS), event, sequences of instructions, loops, and conditional instructions.

5 In this chapter, we mean by *algorithmics* (in French "algorithmique," as used in curricula) the use, creation, and study of algorithms.

6 In France, Technology is a school subject centred on the study, understanding, and design of technical objects.

The two first points, centred on problem solving using computer programs, can be linked with Wing's (2006, 2010) view on CT, or with the Computational Problem-Solving Practices characterized by Weintrop et al. (2016), making explicit some CS notions to be taught (in the third point).

The official resources cited above state that the programming language must be a block language (like Scratch), and there is a strong orientation towards programming. Although accompanying documents for teachers indicate that *unplugged* activities could be a first step for introducing students to the notion of algorithmics, such activities are presented mostly as a steppingstone to the usage of software.

A new high school curricular reform started in 2019, changing completely the structure of the general paths, introducing a choice of "specialties" by students which include Mathematics and a newly created subject, NSI (Numérique et Sciences Informatiques, Digital technology and Computer Sciences). The Mathematics curriculum still contains algorithmics, now presented both as a specific theme ("algorithmics and programming") and with specific examples of algorithms across other themes. Under the theme "algorithmics and programming", it is explicitly indicated that the "algorithmic approach is, since its origins, an essential component of mathematical activity" (MEN, 2019, p. 15, our translation). The main thrust of this part of the curriculum is still to introduce mathematics that can be carried out by a machine through designing algorithms and programming; however, there is no explicit mention of CT. The NSI curriculum is based on four concepts, considered as the pillars of CS ("informatique"): data, algorithms, languages, and machines.

3. Exploring Computer Science and Mathematics: perspectives on CT and content

In this section, to carry on our exploration about how CT and digital tools transform contents, we examine mathematical thinking (MT) and CT from the point of view of the disciplines Mathematics and CS with respect to their links, differences, and mutual contributions (Modeste, 2012b). In the previous section, we have shown that despite questions and debates about the nature of CT and its place and role in mathematics, there is an international curricular movement to integrate CT, explicitly or implicitly—particularly in mathematics education. The three examples we developed, and the existence of international recommendations (e.g., by the OECD), reveal common elements to this integration, such as the introduction of programming for solving problems across mathematical domains and the development of specific skills related to CT as conceptualized in the literature. This said, the way that CT has been introduced in mathematics curricula can also differ greatly from one country to the other (e.g., through the development and application of "coding skills" in Ontario, "CT components" in Australia, or "algorithmics and programming" in France). At the same time, we observe the development of CT in some countries through a specific and distinct curriculum topic (e.g., Computing, Digital Technologies, or CS), or through a collection of CS concepts, methods, and tools that cannot be reduced to CT.

Lodi and Martini (2021) recently demonstrated the importance of considering CT as the thinking of CS, in a way that respects both Wing's and Papert's views on CT and, moreover, that shows the complementarity and coherence of their contributions:

> CT is not a discipline *per se*. CT must be understood in its proper context, which is that of CS. As it is the case for all sciences, CT is *also* transversal to other disciplines, but cannot be reduced to this role. (pp. 884-885)

In this way, just as MT is considered as the thinking developed in mathematics, we will consider CT as the thinking developed in CS. In this sense, we follow Donald Knuth (1985), pioneer of CS, who discussed the links between MT and "Algorithmic Thinking," which he considered as the thinking of CS (which he believed should be named "Algorithmics", a name focusing more on the role of algorithms and less on the machines processing them).

CS as an academic discipline and school subject has grown in the last decades. Some countries have had CS in their secondary school curriculum since the 1980s; others attempted to introduce it in the 1980s, abandoned it for the teaching of digital tools and the use of technologies in other disciplines, and eventually brought it back in the curriculum in the 2010s. The revival of CS aligns with the growth of literature on CS education, including discussions on the relevance of a mandatory CS education for all (e.g., Passey, 2017), reports on the state of CS in school education across countries (e.g., Hubwieser et al., 2015), and curricular frameworks that seek to increase opportunities to learn CS in schools (e.g., K12 Computer Science Framework, 2016, developed in the USA). Such growth of CS in school curricula warrants questioning its links to other (traditional) school topics, and particularly the very closely related topic of mathematics (Modeste, 2015).

There are interesting examples of curricula that encourage cross-curricular links between CS and mathematics. As one such example, we described the theme of "algorithmics and programming" in France, which was recently (2019) integrated in both the Mathematics and Technologies or CS curricula for Grades 7 to 12, with an explicit encouragement of co-teaching and cross-curricular connections. Another example is the recently (2016–2018) revised technologies and mathematics curricula in British Columbia (Canada), which have placed optional Grade 11 and 12 CS courses in the curricular area of Mathematics, explicitly emphasizing the connections between the two domains (British Columbia, n.d.). There are also countries whose distinct CS curricula are intended to be integrated in cross-curricular manners. For instance, New Zealand's 2020 Digital Technologies curriculum for Grades 1 to 13, which includes the development of CS concepts and skills, also indicates that these should be developed in other subject areas (including Mathematics), at least in primary school (New Zealand Ministry of Education, 2022). Such examples further encourage questioning the links between CS and mathematics.

To explore the content transformations that can be brought by digital technologies and CS in mathematics, and to widen our view on the related opportunities or necessities for mathematics education, we examine the links between mathematics and CS, and MT and CT, building on what was addressed in Section 2.1: in what ways does CS generate (new) needs and questions in mathematics, and how does it bring new points of view in mathematics? In brief, we further explore different links between thinking like a computer scientist and thinking like a mathematician.

3.1. Contents at the interface of mathematics and CS

The proximity between mathematics and CS has led to the growth of fields at the interface of the two disciplines, which share points of view from the two disciplines and involve both MT and CT. As examples of such fields, we can mention:

- Cryptography, which involves algebra and arithmetic, but also algorithmics, data structures, information theory, complexity theory, and very important proof and verification issues.

- Operations research, which for industrial problems involves discrete mathematics, optimization of functions, algorithmics, complexity theory, and theoretical CS.
- Image processing and 3D-vizualization, which involves geometry, analysis, statistics, data processing, algorithmics, computer graphics, and compression of data.
- Artificial intelligence, which draws on the interplay of areas such as statistics, big data analysis, management and processing, continuous and discrete optimization, and logics and automatic reasoning.

Computational branches of many disciplines, incorporating both MT and CT, have also emerged, including computational physics, computational biology, computational finance, computational genomics, or digital humanities, which rely heavily on modelling and simulation.

Such contents tend to enter in curricula as good examples of STEM topics and to motivate mathematics with its connections to modern subjects. For instance, in France, as in many other countries, cryptography is mentioned at many grade levels as a potential topic for projects involving both mathematics and CS, starting with the implementation and study of Caesar cipher in middle school. Artificial intelligence, big data analysis, and simulations through agent-based modelling were identified among the rich possibilities for CT integration in K-12 STEM by a recent review of curricular strategies and research themes on the topic (Wiebe et al., 2020). Chapter XX (Biehler et al.) of this handbook also shows some examples of introduction of "big data" and AI methods as a subject in mathematics. These contents, originated in new fields appearing with the development of CS, do not directly stem from the use of digital tools in classic mathematics school contents, but clearly require attention in research on CT in mathematics. The framework of CT practices by Weintrop et al. (2016) (Figure 6, Section 2.1) was also inspired by the emergence of computational branches in the disciplines of science and mathematics; we point out the importance of taking into account CS itself too.

A final significant point about the interface between CS and mathematics is that some theoretical foundations and important concepts are shared, like variables, functions, and algorithms. Even if these concepts can be used differently in the two disciplines, they refer to similar concepts from theoretical and logical points of view, as part of the formal dimension common to mathematics and CS. Hence, taking into account CS and its epistemology can be helpful to understand how mathematics contents can be impacted and enriched.

3.2. Mathematical needs for CS as another type of content transformation

CS not only affords (or requires) changing the view on classical mathematics but also asks new questions to and about mathematics, and fosters the development of specific areas—for instance, discrete mathematics developed over the last decades mostly as an answer to mathematical needs for CS (see Ouvrier-Buffet et al., 2018).

An important example is the study of algorithms, and in particular algorithmic complexity (i.e., the study of the number of steps needed to execute an algorithm as a function of the size of the input). This function can be studied exactly or asymptotically. The notion of complexity enables the comparison of different algorithms solving the same problem or the comparison of problems in terms of the complexity of the algorithms required to solve them. The study of algorithmic complexity asks new questions regarding the study of sequences and the analysis of their asymptotic behaviour, which concerns mathematical analysis and combinatorics. This kind of study is accessible to high school students (the related mathematics and CS contents are already present in the curricula in some countries). In fact, very basic algorithms existing in many high

school curricula (e.g., sorting algorithms as the one in Example 2, or algorithms for processing texts) can raise very interesting mathematical questions in combinatorics and motivate the mathematical study of sequences. For example, the simple case of algorithms for searching in an ordered list generates questions regarding sequences, optimization, graphs and trees, and proving properties of algorithms (Meyer & Modeste, 2018, 2020). The theory of algorithmic complexity has developed as a specific area of CS that uses a lot of mathematics and shares a lot with mathematical areas. Elements of algorithmic complexity are typically taught at the outset of university and serve to motivate mathematics taught to CS majors; algorithmic complexity also starts to appear in upper secondary school in some countries.

The notion of proof, already an important part of mathematics, is equally important in CS, which necessitates proofs of particular types. For example, in algorithmics, the nature of the proofs involved are similar to mathematical proofs, but their type is often particular to the proofs made in discrete mathematics or combinatorics (namely, constructive proofs using induction or enumerations, or identifying invariants) due to the nature of the objects of CS. The example of the sorting algorithm activity described in Section 3.4 illustrates the nature of proving issues in algorithmics. For more details about these issues, see the examples in Durand-Guerrier et al. (2019), Meyer and Modeste (2018, 2020), and Modeste and Ouvrier-Buffet (2011).

This points out an important dimension to be integrated in the study of CT in mathematics education, the content transformations engendered by the need of mathematics in CS and for digital technologies, which is complementary to the more documented and studied transformation of contents produced by the use of digital technologies as tools in mathematics.

3.3. CS affording and necessitating a new point of view on common mathematics

Writing a computer program involving some mathematical concepts “provides a broad canvas on which the learner can sketch half-understood ideas, and assemble on the screen a semi-concrete image of the mathematical structures he or she is building intellectually” (Noss & Hoyles, 1996, p. 55). It also may afford or necessitate an understanding of mathematical concepts and structures that is different from a more common one used in a paper-and-pencil environment—a computational transposition (Balacheff, 1993; Balacheff & Kaput, 1996).

As an illustration, let’s take the representation of numbers. In programming languages, integers typically are represented in base two, which encourages work on number bases—and on base two in particular—in mathematics courses. When it comes to the representation of real numbers on computers, more issues arise. The classical representation is as floating-point numbers, which are rational numbers that can be written as a product of an integer part of significant digits and an exponent part (both coded with a finite number of bits)[7]. This is very different from the common decimal representation of numbers in mathematics (with a definite number of digits before and after the decimal point): The principal difference concerns the density of the sets of numbers, as the floating-point numbers are more dense the closer one gets to zero (and more sparse the farther one gets from zero), which produces issues related to rounding and the precision of computations. Another big difference comes from the use of a base two representation of these numbers. If we just consider the real numbers that can be written with a finite number of digits after the decimal point (i.e., a subset of decimal numbers), there are still many (e.g., 1/10) that cannot be represented as floating-point numbers in base two (just like 1/3 cannot be in base 10).

7 For more details, see “Floating-Point Arithmetic” (2022).

The issues raised by coding and representing numbers in CS can generate very interesting problems and reflections for mathematics curricula in upper high school or university.

A similar discussion could be developed concerning other mathematical topics; for example, let's consider Example 1 in Section 1 (the use of Scratch to explore shapes and patterns), which represents a well-known activity currently in mathematics education, grounded on the legacy of "Turtle Geometry" (Papert, 1980). A related classical problem given in Scratch, in elementary or secondary school, is to draw a regular polygon for a given number of n sides. This problem can be used to introduce, for instance, the "repeat *x* times" loop, identifying that some similar instructions need to be repeated a variable number of *n* times, with some parameters depending on n. A solution to the problem is illustrated in the program shown in Figure 7, with the output of its execution for n = 13.

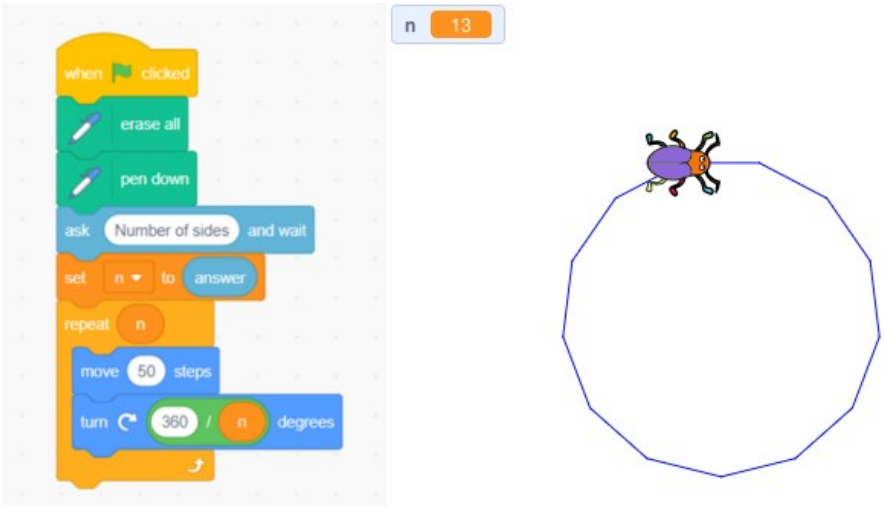

Fig. 7 A Scratch program for drawing a regular polygon with n sides, and the effect of its execution for *n* = 13.

We see in this example that the length of the sides of the polygon is fixed, while the radius of the polygon grows with n. Moreover, the point of view on regular polygons involved in the Scratch program is not exactly the point of view involved classically in mathematics. Classically, a regular polygon is defined in school as a polygon with equal sides and equal internal angles, which can be obtained by placing its vertices regularly on a circle (in this view, it is possible, though not obvious, to show that the internal angles of an *n*-sided polygon are 180 (*n*-2)/*n* degrees). In the Scratch program, the point of view is different, as the regular polygon is seen as a figure obtained by tracing a broken line composed of line segments of the same length and "turning" (clockwise or anticlockwise) an identical angle between each of them; the angle expressed here is not the internal angle of the polygon but rather its supplement (i.e., the angle between two consecutive sides oriented in the same way; for a *n*-sided polygon, 360/*n* degrees– 360 degrees is an invariant quantity for the full turn).

This new point of view is principally linked with the way geometrical position, orientation, and movement are described in Scratch (formerly referred to as "Turtle Geometry"). The sprite (the character executing the program, in this case the beetle in Figure 7) has coordinates and a direction. When the programmer wants to make the sprite move, it can move to specified coordinates or be reoriented to a specified direction; it can also move a specified number of steps following its direction or turn (left or right, staying at the same place) a specified angle. Hence,

the new point of view on plane geometry afforded by Scratch changes the way geometrical figures and transformations are considered (for another example, see Crisci et al., 2022, about symmetry), leading to rich possibilities for MT and the question (perhaps need) of its incorporation in mathematics curricula.

Hence, the development of CT, generated using CS or digital technologies in classical school mathematics, may not only change the way we deal with mathematical concepts. By the fact it can change the way one thinks about them, it can transform the concepts themselves. In addition to investigate how the mathematical activity may be transformed by CT, this argues for further research on how some mathematical concepts may be transformed by CT.

3.4. An example of these various relations between mathematics and CS

We return now to Example 2 presented in Section 1, in which middle school students (upper primary and junior secondary) can be invited to enact and reflect on the bubble sort algorithm[8]. Starting from the left, pairwise sorts are carried out with only one decision to be made. The larger number in each pairwise sort moves to the right and the next pair is sorted. When all pairwise sorts are completed, passing along the line, the process starts again at the beginning. At each pass, the number arrived at the rightmost position won't move anymore and is no longer included in the next passes. The process stops when no further passes are needed (i.e., when no number has moved during a pass), which implies all numbers are correctly sorted. In the classroom, this activity raises some interesting questions to investigate at the interface of CS and mathematics: Does the algorithm always work (for any initial placement of the numbers in the line)? Does it work for more than 10 numbers? For any number of numbers? If yes, why? How many comparisons do we need to sort 11 (or 12 or 100) numbers? Can we generalize to any number of numbers? In this subsection, we further reflect on the bubble sort activity as an example concerning CS and mathematics, and how it can contribute to CT and MT.

Order is a classical concept of mathematics, and sorting is a classical issue in CS, and sorting algorithms are traditionally taught at the beginning of CS studies (see, for instance, Cormen et al.'s (2009) classical handbook for the introduction of algorithms). In the activity described above, the algorithm being followed (the bubble sort), which is unambiguous, can be generalized for any list of numbers, and can be made into a computer program. Enacting and studying this algorithm, for students, comes under CT but also involves elements that can be associated with MT. From an expert's point of view, the questions above relate to the field of Algorithmics, and are classical in CS (and can be considered as typical of CT):

- proving the algorithm will end for any list of numbers (termination); and it will produce a sorted list of the initial integers (correction),
- studying its complexity (i.e., estimating the number of steps needed to execute the algorithm for a given size of list).

After observing several enactments of the algorithm, as mentioned in Section 1, a question can be posed to students asking how many *passes* will be needed to ascertain that any 10 numbers will be sorted successfully. This kind of question can trigger MT with a view to developing argumentation, as a preparation for proving. It is distinct, although not without links, from questions about how to write an algorithm that will give clear instructions for a bubble sort to be executed.

8 This activity can be considered as "CS Unplugged," as developed by Bell et al. (1998), and presented later.

Below are excerpts of three sample solutions, adapted with minor editing, from students in two classes where the bubble sort activity had been carried out.

- *Solution 1 (Year 6 student, age 11–12): Starting with a smaller set of numbers*
  I made it into a simple problem to start with. If there were just two numbers, they might be in order already or they might need to shift. Therefore, one student would be sure to "sort out" any two numbers into the correct order. If there were three numbers in a line, two students would be enough to be sure that the numbers ended up in the correct order. So, my answer for 10 students in the line is that nine students would be sufficient to completely sort the 10 numbers.

- *Solution 2 (Year 8 student, age 13–14): A worst-case scenario, for 10 numbers*
  We can represent the worst possible case by having the numbers 1 to 10 lined up with 10 on the left and 1 on the far right. Student 1 comes out and shifts the 10 to the top, after doing swaps at every step. Student 2 does the same for 9. Student 3 for 8. Then Student 4 for 7; Student 5 for 6; Student 6 for 5; Student 7 for 4; Student 8 for 3; and Student 9 for 2. That's all.

- *A comment (Year 6 student):*
  If you were very lucky and all the numbers were in the correct order to start with, we would still need one student to carry out that check. We might be able to see this by looking, but the "sorting students" don't know this.

These solutions exhibit different strategies discussed by Pólya (1970) for proving and justifying. The first solution starts by looking at a simpler problem and reaches a conclusion using inductive reasoning (although some arguments are missing, we can notice that it is possible to prove the statement via mathematical induction). The second solution creates a worst-case scenario where all the numbers are ordered in reverse to start with (the students do not completely justify that it is the worst case). These two justifications are concerned with generalizing. The final comment by a Year 6 student shows how clearly the concept of worst- and best-case situation has been understood in this context.

These solutions and comment use justifying, convincing, and generalizing as key features of MT (see Jeffcoat et al., 2004). Engaging in CT activities, like the bubble sort activity, can provide contexts in which young students can be invited to justify their thinking and to become familiar, as Coles (2005) suggests, with the process of creating convincing mathematical arguments.

3.5. Synthesis

To conclude this section, we propose a diagram (Figure 8) that synthesizes relationships between mathematics, CS, MT, and CT. It widens the view on how mathematical content is transformed by CS and digital technologies. We position in this diagram the discussions from sections 2 and 3. Finally, this also raises the question of how mathematical contents (objects, notions, techniques, skills, mathematical fields...) can support the development and mutual enrichment of MT and CT.

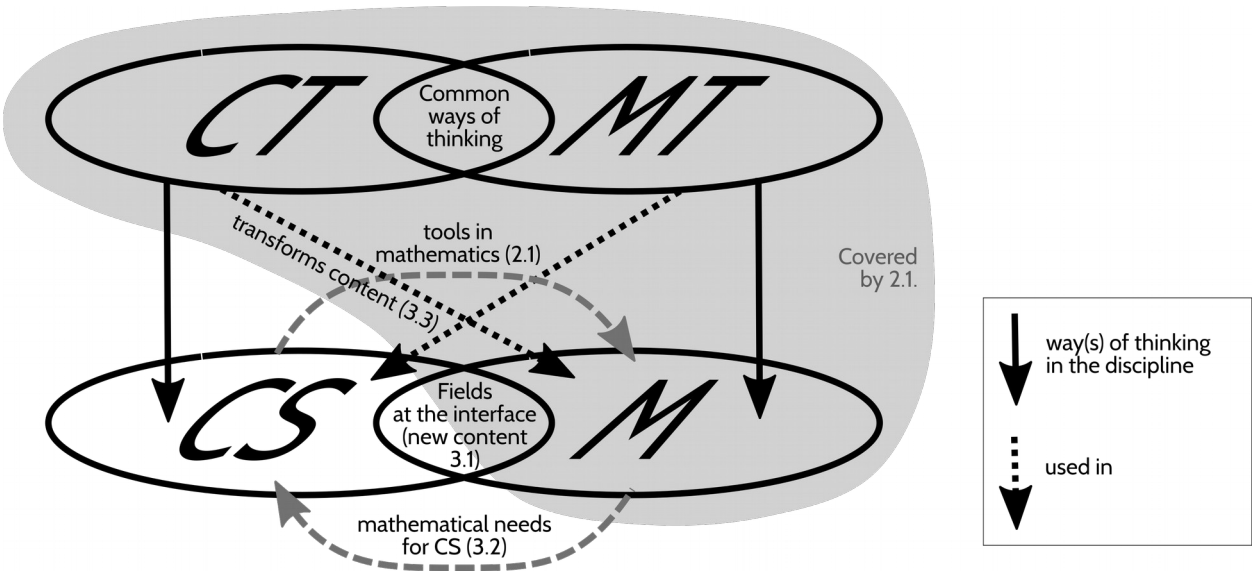

Fig. 8 A model of mathematics (M), CS, MT, and CT, highlighting mutual contributions as discussed in sections 2.1, 3.1, 3.2, and 3.3.

4. Implementation of prescribed curricula and diffusion of transformed content

In the previous sections, we explored how the integration of CT in mathematics education can transform content and outlined international curricular revision movements that may serve to support this process. In this section, we shift our focus to issues related to the actual implementation of curricula by teachers in order to understand what the obstacles and difficulties are for the diffusion of identified mathematics content transformations in teaching and learning.

A systematic review of literature published between 2006 and 2017 investigating relationships between mathematics and CT demonstrated through didactic activities (Barcelos et al., 2018) revealed that most of studies focused on learning opportunities for students. Only three of 42 reviewed papers targeted teacher education from the perspective of the diffusion of CT in mathematics education. Research in the past 5 years, however, has considered the teacher's role in embedding CT in mathematics lessons. This could be explained partially by the history of CT outlined in Section 2. In particular, the initial focus on opportunities aligns with the nascent phase of the CT movement led by Wing (2006) during which time there was an interest in developing curricular revisions, making work on teacher education more essential. Moreover, research has observed significant differences between the design of curricular materials integrating CT and mathematics and their implementation in practice (e.g., Benton et al., 2017; Crisci, 2020; Israel & Lash, 2020; Nijimbere, 2015). In this section, we examine research related to the actual implementation of CT[9] in mathematics classrooms that, in its turn, determines the way the mathematical content proposed to students is actually transformed by teachers, highlighting factors that shape teachers' practices, as well as perspectives for the design of supports for teachers (e.g., teacher resources and development programs). Given the themes of the handbook, we focus specifically on the design and implementation of (digital) resources.

9 As illustrated in Section 2, the chosen definition of CT can vary across the research cited in this section. We describe this in more detail when relevant; otherwise, we speak about general issues and perspectives that we expect could apply across definitions.

4.1. Factors influencing the implementation of CT and actual content transformation in mathematics classrooms

Existing studies have identified several factors that can influence mathematics teachers' implementation of CT and thus actual content transformations that may happen in the mathematics class, such as different kinds of teachers' knowledge (e.g., Benton et al., 2017; Duncan et al., 2017; Rich et al., 2019), teachers' attitudes towards teaching mathematics (e.g., Crisci, 2020) and CT (e.g., Yadav et al., 2014), and teachers' (mis)conceptions related to CT (e.g., Cabrera, 2019; Geraniou & Hodgen, 2022; Rich et al., 2019; Sands et al., 2018).

Much of the above research documents teachers' difficulties integrating mathematical content and CS/CT ideas, including those relating to the teaching practice in general and to teaching CT and mathematics specifically. In Benton et al. (2017), for instance, examples of the first type of difficulties include fitting new interventions into an already crowded timetable; a lack of explicit links between proposed learning activities and existing curricula; and managing differences among students in terms of their technological skills, support needs, and confidence. Israel and Lash (2020) also pointed out that the pressure on teachers to cover the mathematics content to prepare students for district and state assessments may impede teachers' ability to integrate CT in their classes.

Difficulties of the second type, in comparison, include making and explaining links between an algorithm and certain mathematical concepts, finding errors in students' scripts and strategies for solving problems involving CT and MT, and effectively anticipating and dealing with different types of students' responses and misconceptions related to CT. According to Benton et al. (2017), the situation can vary depending on teachers' capacities to articulate computational and mathematical learning goals, their confidence in using technologies, their experience, and their ability to managing students with different levels of attainments. We suggest that the integration of CT in mathematics classrooms depends in particular on teachers' knowledge about CT and MT content, as well as their corresponding pedagogical content knowledge (Shulman, 1986).

Several studies exhibit cases in which existing teaching knowledge can constrain integration of CT in mathematics education. Rich et al. (2020) found that elementary teachers reflecting on the meaning of six CT practices (abstraction, algorithmic thinking, automation, debugging, decomposition, and generalization) tended to refer to their knowledge of terminology used in mathematics teaching instruction and curricular standards. For instance, when speaking about "decomposition," teachers provided an example of decomposition of numbers without considering the decomposition of a problem, and they associated the term "automation" with the memorization of basic mathematics facts and students' automatic reproduction of answers. Similar observations were made by Duncan et al. (2017), who mentioned, for example, teachers' difficulties in using the term "sorting", which in primary school typically refers to categorizing rather than "ordering" items.

Other factors that impact implementation include teachers' perceptions, beliefs, and attitudes about CT. Research shows that teachers hold a variety of conceptions of CT (e.g., CT as doing mathematics, CT as using technologies, CT as programming, or CT as problem solving), which can differ from CT definitions described in the academic literature (Corradini et al., 2017; Garvin et al., 2019; Sands et al., 2018). Varying conceptions of CT also have been observed among mathematics teacher educators, which could further influence teachers' conceptions. Geraniou and Hodgen (2022) claim that teacher educators' understanding of CT can vary depending on

how they see the links between CT and MT, and the role of the application (or non-application) of digital technology in CT. The authors contrast two views of CT: one referring to the use of digital technology for outsourcing some (possibly tedious) parts of mathematical work, and another that focuses on the processes of analyzing, generalizing, and abstracting involved in CT (not mandatorily related to the usage of technologies).

Other obstacles to the integration of CT in mathematical teaching can be related to the scarcity of infrastructure in schools and teachers' resistance to introduce something new and unknown (Reichert et al., 2020). These aspects seem to persist in the context of integration of CT tools in mathematics classes despite school's increasing level of informatization.

The emerging body of research outlined above is very important from the point of view of possible applications such as the design of resources and teacher development programs that could support mathematics teachers' integration of CT and enacting content transformations in their classrooms. For instance, research findings can inform curricula developers about the use of specific terms, relevant definitions, and examples to avoid ambiguity and possible tensions between "new" terminology and that already used by teachers. Design of professional development programs and resources should integrate (research-based) epistemological elements related to the compatibility and complementarity between CT and mathematics. Furthermore, relationships between digital technologies and CT should be clarified to avoid the possible misconception that associates any usage of technologies with CT practices.

We note that the above discussion relates primarily to the school context (Grades 1-12) and may not necessarily apply in the same way at the university level, where there is a deeper understanding of CT and its links to mathematics (Buteau et al., 2019). Nevertheless, university mathematics professors may still face certain institutional constraints if attempting to integrate CT in existing mathematics courses (e.g., limited time, already crowded curricula, students' prerequisites, lack of human resources; Broley, 2015).

4.2. Supports for the diffusion and appropriation of content transformations by teachers

4.2.1. Resources as a tool for the diffusion of content transformations in mathematics teaching and learning

Developing resources that inform teachers' implementation of CT is one way to support the diffusion of content transformations, involving CT activities and experiences in mathematics education, across a wide population of teachers working in different contexts. However, research on resource design (e.g., Aldon et al., 2017; Perrin-Glorian, 2011; Tempier, 2016) has shown that teachers interacting with the same resource can interpret it in different ways than intended by the task designers. Chapter XXX (Mariotti et al.) of this handbook also reports issues of resources' adaptations by a teacher that prevents achieving the goals. In case of resources aimed at integration of CT in a mathematical class, the question of possible adaptations of resources by teachers and the corresponding consequences on students' learning becomes especially important. Teachers' lack of expertise and experience with CT can impact their interactions with resources and, as a result, the design end implementation of the lessons developed on their base.

Rafalska (2019) studied the elements a resource aimed at integrating CT in mathematics classrooms should contain to support the intended learning during its implementation in different contexts. The author designed and investigated a resource for primary school teachers involving

a sequence of unplugged games where students “re-invent” sorting algorithms (Rafalska, 2022). The proposed task, as the earlier Example 2, shows how mathematical content and students’ mathematical activities can be enriched by the introduction of and reflection on the concepts related to CS. However, findings confirm that a resource in the form of a lesson plan alone, even when designed in collaboration by researchers and teachers in a particular context, is not robust enough to prevent adaptations made by teachers (working in other contexts) that lead to replacement of the initial task by another, the goals of which are not consistent with those that task designers expected. Students’ activities that emerged in certain classes after the adaptations of the resource by the teachers focused on learning goals that differed from the ones intended by the resource, especially concerning the goals related to CT and mathematics (that were in core of the proposed tasks). Moreover, the activities proposed by one of the teachers on the base of the resource did not achieve expected learning goals either in CT or in mathematics. This research be resource’s CT and mathematics goals, the resource should include more than a lesson plan—for instance, an a priori didactic analysis of the situations supported by the resource, descriptions of the knowledge at stake, theoretical and epistemological aspects of CT, and links to existing mathematics curricula.

Results of design-based studies addressing the gap between the design and implementation of resources can also inform resource design through the identification and/or implementation of effective teaching strategies for integrating CT in mathematics classrooms. For example, Crisci (2020) analyzed cases where implementations were aligned with the expectations of researchers and authors of resources, highlighting the important role of teachers verbalizing (using oral registers to make the link between different ostensive registers) and de-contextualizing (passing from extra-mathematical context vocabulary to mathematical vocabulary) in supporting the usage of resources in coherence with learning goals intended by their authors. In their design of resources, Benton et al. (2017) discussed the importance of off-computer (“unplugged”) activities, as well as the use of different representations simultaneously, for students’ engagement and deep learning.

More generally, research has identified many effective teaching approaches that could help integrate CT or CS in mathematics education, which can both inform the design of resources and serve as useful resources themselves. For instance, Curzon et al. (2018) discuss and exemplify diverse constructivist approaches for inventing or discovering CS concepts, such as teaching through analogy and storytelling, embedding concepts in contexts, and utilizing unplugged computing activities. Other teaching strategies include scaffolding techniques such as the Use-Modify-Create or Predict-Run-Investigate-Modify-Make learning models, whereby students move back-and-forth between users and developers (Caspersen, 2018; Waite & Grover, 2020). Such strategies can be found in emerging resources for integrating CT in mathematics education (see, for example, Gadanidis, 2020). Coming from a more constructivist tradition, Resnick (2014) used a “Projects, Peers, Passion, and Play” approach to engage students in meaningful projects with the aim of developing the skills needed to use CT to creatively solve problems. Broley et al. (2022b) explore and document the effectiveness of a project-based approach in university mathematics, where the projects integrate the affordances of CT for mathematical practices (as described in Section 2.1). Key features of teaching to support such a project-based approach have also been discussed (e.g., by Broley et al., 2022a; Buteau et al., 2022).

4.2.2. Professional development programs to support appropriation of content transformations by teachers

Research on teacher education to support the integration of CT in school classrooms is growing. Several case studies have emerged that present and discuss examples of how to support mathematics teachers. For pre-service teachers, this includes case studies of courses that focus specifically on integrating CT in mathematics education (e.g., Cendros Araujo et al., 2019; Gadanidis et al., 2017), or of existing methods courses (e.g., Bouck et al., 2021) and/or practicum placements (e.g., Suters, 2021) to which pedagogical issues related to CT and mathematics have been added. There are also case studies of introducing CT in in-service training for mathematics teachers, including continuing education courses (e.g., Reichert et al., 2020), summer professional and curricular development programs (e.g., Kelter et al., 2021), and professional development (PD) sessions coupled with continuous instructional coaching on lesson planning (e.g., Israel & Lash, 2020).

These studies emphasize the importance of providing content-specific education to teachers (in our case, teaching CT that is specific to mathematics). The emerging body of research in this domain focused mostly on how to improve teachers' confidence with CT, understandings of CT and its value in mathematics, and practices for productively integrating CT into mathematics classrooms.

For instance, several studies present different pedagogical approaches used in teacher education programs focusing on what CT is and how it can be integrated in mathematics classes. Kelter et al. (2021) claim that a "constructionist co-design" between teachers and researchers helps to support curriculum and teacher PD. They reported about the positive effects of the course on high school science teachers' understanding of the ways CT can support students' experience and deepen science learning as well as on teachers' confidence with CT. However, the researchers pointed out the tensions that can arise in trying to accomplish both goals (to develop teachers CT skills and engage them in design of instructional units) simultaneously in limited time. In particular, the teachers with less computational experience tended to leave the development of computational tools to their partners; as a result, this limited their opportunities to improve their own CT skills. Besides, a co-design style used by teachers in such type of PD programs seems to affect teachers' implementation of lessons that articulate computational and mathematical goals. In particular, teachers with limited skills in programming tend to avoid explanations concerning debugging programs or, as was observed in Reichert et al.'s (2020) study, tend to call on CS assistance.

Several studies emphasize the importance of building teachers' confidence in and knowledge of CT for mathematics before inviting them to apply that knowledge for teaching. For instance, Cendros Araujo et al. (2019) recommend that training should start with action, "allowing [teachers] to tinker, play, and experiment with programmable technology to develop their confidence" (p. 225) and then introduce them to features of the technology before inviting them to apply it to the subject of teaching mathematics. Teaching approaches such as "Action – Learn and teach – Extend, apply and consolidate – Reflect" have positive effects on secondary school pre-service teachers' understanding of CT and its integration in the context of mathematics.

Studies on the effects of teacher education programs are mostly represented by case studies based on the qualitative analysis of data collected by researchers who usually serve as tutors/providers of such programs. These studies are characterized by use of data sources very

often collected during education programs, usually with a small sample size and represented by teachers' written or oral discourses (e.g., essays, written reflection assignments, online discussions, interviews) about their understanding of CT and its integration in the curriculum. Thus, it is not surprising that teachers' words reflect the content proposed in the PD programs (definitions, examples of implementation, etc.). More research is needed to examine the extent to which these approaches influence teachers' knowledge and practice beyond the PD course as well as the effects of such teaching on students' learning and actual content transformation.

5. Conclusions and perspectives

The use of transformative digital tools has taken different forms, for example, by introducing coding, programming or algorithmics, initially in stand-alone computer science curricula, but increasingly within mathematics curricula, and through trans-disciplinary approaches. This chapter has examined how these differing digital tools play a role in advancing the teaching and learning of mathematics and developing students' mathematical agency and identity. This chapter has also explored the links between mathematics and CS, and MT and CT, showing that CS generates new needs and questions in mathematics, and how CS brings new points of view in mathematics.

The growing use of digital tools has drawn attention to the relationship between mathematical thinking and computational thinking. As this chapter has shown, the use of digital technologies cuts across content, teaching practices and student learning. Digital technologies allow new areas of content to be considered, such as algorithmisation, experimentations, approximation, conjecture testing, visualisation and modelling. We also note that these content areas also transform aspects of mathematical thinking for both teachers and students. Using digital resources, students can become creators of and partners in new ways of conceptualising and solving problems. And in these contexts, teachers of mathematics must adapt to new ways of thinking like computer scientists, giving rise to new challenges for teacher education and teacher professional learning.

This chapter has pointed to an emerging international conversation about CT among various stakeholders which is now being increasingly integrated in school classrooms and curricula worldwide. In our view, this chapter illustrates that it is not simply a trend; we argue that it constitutes a change that is likely to continue in the long term.

In the senior secondary years, many countries already have elective courses for Computer Science (CS), programming, applied computing, and algorithmics to provide a transition to university courses in mathematics, CS, and related disciplines where computational practices are already incorporated. However, including CT in basic education for all students is a relatively new aspect of many countries' national curricula. A key decision for policy makers is where to make the split between compulsory education and later years of schooling where greater opportunities for choice and specialization can be provided.

At this point in time, there is no clear consensus on how to integrate CT in compulsory curricula. For instance, CT is now included, explicitly or implicitly, in the subject area of mathematics in many jurisdictions, such as Australia, Ontario (Canada), and France. But even in these countries, we can see that CT is present in different ways. This, as well as the variety of definitions considered in educational research, suggest that discussions about the nature of CT and its contributions to mathematics (and vice versa) are still needed. We expect that a multiplicity of

understandings of CT will nevertheless remain, with different purposes in both research and practice. It is therefore important that researchers and policy makers be clear about which definition and scope of CT they use and how this impacts the way they present results or ideas to researchers or teachers. There are other critical tasks in integrating CT in mathematics curricula, such as the development of sequences of learning that are coherent with the mathematical contents and with what students are learning in other subject areas (e.g., digital technologies, CS, or other areas where CT has been integrated). It also will be necessary to think about what (additional) mathematics is needed for learning CS and to contribute to a deeper study of digital technologies.

Even if CT is integrated in prescribed curricula and there are corresponding resources with "good" tasks, enacted curricula, and implementation of resources by teachers can differ greatly from what policy makers and task designers expect. High-quality resources are needed, that consider teachers' existing practices and previous education, to support the regular inclusion of CT in mathematics teaching. Research on the design of resources and CT-related PD programs and their impact on teachers' understandings and practice is also needed.

We finish the chapter by proposing three future challenges for mathematics education. First is the continued challenge of showing how CT can complement and enhance the teaching and learning of mathematics for all students, and how this can be integrated in mathematics curricula and classrooms at all levels of education. As school curricula related to the study of CS are under development, a second challenge appears: identifying opportunities for CS to interact with mathematics and balancing the teaching of the two disciplines, which must take into account the respective contributions of one to the other and their synergy. Finally, there is the challenge of developing research-informed PD programs and resources for mathematics teachers to integrate CT in their teaching, without which we cannot expect meaningful integration on a broad scale.

References

Aldon, G., Front, M., & Gardes, M-L. (2017). Des intentions de l'auteur aux usages en classe, première réflexion sur la cohérence des usages d'une ressource [Between designing and use, how to address the question of resource's consistency?]. *Éducation & Didactique*, *11*(3), 9–30. https://doi.org/10.4000/educationdidactique.2810

Australian Curriculum Assessment and Reporting Authority (ACARA). (2022). *The Australian curriculum version 9.0: Mathematics*. https://v9.australiancurriculum.edu.au/

Balacheff, N. (1993). La transposition informatique, un nouveau problème pour la didactique. In M. Artigue, N. Balacheff, R. Gras, C. Laborde, & P. Tavignot (Eds.), *Vingt ans de didactique des mathématiques en France, 15-17 juin 1993* (pp. 364–370). La Pensée Sauvage. https://telearn.archives-ouvertes.fr/hal-00190646/document

Balacheff, N., & Kaput, J. J. (1996). Computer-based learning environments in mathematics. In A. J. Bishop, K. Clements, C. Keitel, J. Kilpatrick, & C. Laborde (Eds.), *International handbook of mathematics education* (pp. 469-501). Springer. https://doi.org/10.1007/978-94-009-1465-0_14

Barba, L. A. (2016, March 5). *Computational thinking: I do not think it means what you think it means*. Lorena A. Barba Group. https://lorenabarba.com/blog/computational-thinking-i-do-not-think-it-means-what-you-think-it-means/

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.

https://www.jucs.org/jucs_24_7/mathematics_learning_through_computational/jucs_24_07_0815_0845_barcelos.pdf

Bell, T., Witten, I. H., & Fellows, M. (1998). *Computer science unplugged: Off-line activities and games for all ages*. https://classic.csunplugged.org/documents/books/english/unplugged-book-v1.pdf

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,* 115–138. https://doi.org/10.1007/s40751-017-0028-x

Bocconi, S., Chioccariello, A., Dettori, G., Ferrari, A., & Engelhardt, K. (2016). *Developing computational thinking in compulsory education: Implications for policy and practice*. European Commission. http://doi.org/10.2791/792158

Bocconi, S., Chioccariello, A., & Earp, J. (2018). *The Nordic approach to introducing computational thinking and programming in compulsory education*. Nordic@BETT2018 Steering Group. https://www.itd.cnr.it/doc/CompuThinkNordic.pdf

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

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the annual meeting of the American Educational Research Association* (Vol. 1, pp. 1–25). http://scratched.gse.harvard.edu/ct/files/AERA2012.pdf

Brisac, J., Favero, S., Jocaille, L., Joly, L., Marcaille, B., Royoux, G., Terpereau, C., & Terpereau, P. (2021). *Collection TAM: Maths 3e*. Hatier.

British Columbia. (n.d.). *BC's curriculum: Mathematics*. https://curriculum.gov.bc.ca/curriculum/mathematics

Broley, L. (2015). *La programmation informatique dans la recherche et la formation en mathématiques au niveau universitaire* [Master's thesis, Université de Montréal]. Papyrus: Institutional Repository. https://papyrus.bib.umontreal.ca/xmlui/handle/1866/12574

Broley, L., Ablorh, E., Buteau, C., Mgombelo, J., & Muller, E. (2022a). Effective orchestration features of a project-based approach to learning programming for mathematics investigation [Paper presentation]. *Third Conference of the International Network for Didactic Research in University Mathematics*, Hannover, Germany.

Broley, L., Ablorh, E., Buteau, C., Mgombelo, J., & Muller, E. (2022b). Effectiveness of a project-based approach to integrating computing in mathematics. In S. Smith Karunakaran & A. Higgins (Eds.), *Proceedings of the 24th Annual Conference on Research in Undergraduate Mathematics Education* (pp. 72-80). SIGMAA. http://sigmaa.maa.org/rume/RUME24.pdf

Broley, L., Buteau, C., & Muller, E. (2017). (Legitimate peripheral) computational thinking in mathematics. In T. Dooley & G. Gueudet (Eds.), *CERME 10: Proceedings of the Tenth Congress of the European Society for Research in Mathematics Education* (pp. 2515–2522). DCU Institute of Education and ERME. https://hal.archives-ouvertes.fr/CERME10

Broley, L., Caron, F., & Saint-Aubin, Y. (2018). Levels of programming in mathematical research and university mathematics education. *International Journal of Research in*

Undergraduate Mathematics Education, *4*(1), 38–55. https://doi.org/10.1007/s40753-017-0066-1

Buteau, C., Muller, E., Santacruz Rodriguez, M., Mgombelo, J., Sacristan, A. I., & Gueudet, G. (2022). Instrumental orchestration of using programming for authentic mathematics investigation projects. In A. Clark-Wilson, O. Robutti, & N. Sinclair (Eds.), *The mathematics teacher in the digital era* (2nd ed.). Springer. https://doi.org/10.1007/978-3-031-05254-5_1

Buteau, C., Sacristán, A. I., & Muller, E. (2019). Roles and demands for constructionist teaching of computational thinking in university mathematics. *Constructivist Foundations*, *14*(3), 294–309. https://constructivist.info/14/3/294

Cabrera, L. (2019). Teacher preconceptions of computational thinking: A systematic literature review. *Journal of Technology and Teacher Education*, *27*(3), 305–333.

Caspersen, M. E. (2018) Teaching programming. In S. Sentance, E. Barendsen, & C. Schulte (Eds.), *Computer science education: Perspectives on teaching and learning in school* (pp. 109–130). Bloomsbury Academic. http://doi.org/10.5040/9781350057142.ch-009

Cendros Araujo, R., Floyd, L., & Gadanidis, G. (2019). Teacher candidates' key understandings about computational thinking in mathematics and science education. *Journal of Computers in Mathematics and Science Teaching*, *38*(3), 205–229.

Coles, A. (2005). *Proof and insight. Mathematics teaching 190*. Association of Teachers of Mathematics (ATM).

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to algorithms* (3rd ed.). The MIT Press.

Corradini, I., Lodi, M., & Nardelli, E. (2017, August). Conceptions and misconceptions about computational thinking among Italian primary school teachers. *Proceedings of the 2017 ACM Conference on International Computing Education Research* (pp. 136–144). Association for Computing Machinery. https://doi.org/10.1145/3105726.3106194

Crisci, R. (2020). *Etude des conditions de viabilité d'une approche basée sur l'algorithmique et la programmation pour l'apprentissage de la division euclidienne à l'école primaire* [Study on the viability conditions of an algorithmic- and programming-based approach to the learning of Euclidean division in primary school] [Doctoral dissertation, Université Grenoble Alpes]. HAL. https://hal.archives-ouvertes.fr/tel-03116813/

Crisci, R., Dello Iacono, U., & Ferrara Dentice, E. (2022). A digital artefact based on visual programming for the learning of axial symmetry in primary school [Paper presentation]. *Twelfth Congress of the European Society for Research in Mathematics Education (CERME12)*. https://hal.archives-ouvertes.fr/hal-03748428/document

CS4ALLUSA. (2012, December 30). *CS4ALL bubble sort unplugged activity* [Video]. YouTube. https://www.youtube.com/watch?v=glgnCcjgpek

Curzon, P., Bell, T., Waite, J., & Dorling, M. (2019). Computational thinking. In S. A. Fincher & A. V. Robins (Eds.), *The Cambridge handbook of computing education research* (pp. 513–546). Cambridge University Press. https://doi.org/10.1017/9781108654555.018

Curzon, P., McOwan, P. W., Donahue, J., Wright, S., & Marsh, W. (2018). Teaching of concepts. In S. Sentance, E. Barendsen, & C. Schulte (Eds.), *Computer science education: Perspectives on teaching and learning in school* (pp. 91-108). Bloomsbury Academic. http://doi.org/10.5040/9781350057142.ch-008

Dagienė, V., Jevsikova, T., & Stupurienė, G. (2019). Introducing informatics in primary education: Curriculum and teachers' perspectives. In S. N. Pozdnyakov & V. Dagienė

(Eds.), *Informatics in schools: New ideas in school informatics* (pp. 83–94). Springer. https://doi.org/10.1007/978-3-030-33759-9_7

diSessa, A. A. (2018). Computational literacy and "the big picture" concerning computers in mathematics education. *Mathematical Thinking and Learning*, *20*(1), 3–31. https://doi.org/10.1080/10986065.2018.1403544

Duncan, C., Bell, T., & Atlas, J. (2017). What do the teachers think? Introducing computational thinking in the primary school curriculum. In D. Teague & R. Mason (Eds.), *Proceedings of the Nineteenth Australasian Computing Education Conference (ACE 2017)* (pp. 65–74). Association for Computing Machinery. https://doi.org/10.1145/3013499.3013506

Durand-Guerrier, V., Meyer, A., & Modeste, S. (2019). Didactical issues at the interface of mathematics and computer science. In G. Hanna, D. Reid, & de V. Michael (Eds.), *Proof technology in mathematics research and teaching* (pp. 115–138). Springer. https://doi.org/10.1007/978-3-030-28483-1_6

European Mathematical Society. (2011). *Position paper of the European Mathematical Society on the European Commission's contributions to European research*. https://ctuniversitymath.files.wordpress.com/2022/06/european_mathematical_society-2011.pdf

Floating-point arithmetic. (2022, December 3). In *Wikipedia*. https://en.wikipedia.org/wiki/Floating-point_arithmetic

Gadanidis, G. (2017). Five affordances of computational thinking to support elementary mathematics education. *Journal of Computers in Mathematics and Science Teaching*, *36*(2), 143–151.

Gadanidis, G. (2020). *Understanding math + coding*. https://learnx.ca/math/

Gadanidis, G., Cendros, R., Floyd, L., & Namukasa, I. (2017). Computational thinking in mathematics teacher education. *Contemporary Issues in Technology and Teacher Education*, *17*(4), 458–477. https://citejournal.s3.amazonaws.com/wp-content/uploads/v17i4Math3.pdf

Garvin, M., Killen, H., Plane, J., & Weintrop, D. (2019). Primary school teachers' conceptions of computational thinking. *Proceedings of the 50th ACM Technical Symposium on Computer Science Education (SIGCSE19) (pp. 899-905).* https://doi.org/10.1145/3287324.3287376

Geraniou, E., & Hodgen, J. (2022). An exploratory study on mathematics teacher educators' beliefs and understandings about computational thinking. In *Proceedings of the Twelfth Congress of the European Society for Research in Mathematics Education*. European Society for Research in Mathematics Education (ERME). https://hal.science/CERME12/hal-03748436v1

Government of Ontario. (2020). *Curriculum resources: Elementary mathematics*. https://www.dcp.edu.gov.on.ca/en/curriculum/elementary-mathematics

Government of Ontario. (2021a). *Curriculum resources: Grade 9 mathematics*. https://www.dcp.edu.gov.on.ca/en/curriculum/secondary-mathematics/courses/mth1w

Government of Ontario. (2021b, January 18). *The Ontario curriculum: Grades 1 to 8*: *Mathematics*. https://tinyurl.com/ya5yvkne

Grover, S., & Pea, R. (2013). Computational thinking in K-12: A review of the state of the field. *Educational Researcher, 42*(1), 38-43. https://doi.org/10.3102/0013189X12463051

Gueudet, G., Bueno-Ravel, L., Modeste, S., & Trouche, L. (2017). Curriculum in France. A national frame in transition. In D. R. Thompson, M. A. Huntley, & C. Suurtamm (Eds.), *International perspectives on mathematics curriculum* (pp. 41–70). IAP. https://hal.archives-ouvertes.fr/hal-01599059/document

Guzdial, M. (2019). Computing for other disciplines. In S. Fincher & A. Robins (Eds.), *The Cambridge handbook of computing education research* (pp. 584–605). Cambridge University Press. https://doi.org/10.1017/9781108654555.020

Healy, L., & Kynigos, C. (2010). Charting the microworld territory over time: Design and construction in mathematics education. *ZDM*, *42*(1), 63–76. https://doi.org/10.1007/s11858-009-0193-5

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

Hubwieser, P., Giannakos, M. N., Berges, M., Brinda, T., Diethelm, I., Magenheim, J., Pal, J., Jackova, J., & Jasute, E. (2015). A global snapshot of computer science education in K–12 schools. In *Proceedings of the 2015 ITiCSE on Working Group Reports* (pp. 65–83). ACM. https://doi.org/10.1145/2858796.2858799

Israel, M., & Lash, I. (2020). From classroom lessons to exploratory learning progressions: Mathematics + computational thinking. *Interactive Learning Environments*, *28*(3), 362–382. https://doi.org/10.1080/10494820.2019.1674879

Jeffcoat, M., Jones, M., Mansegh, J., Mason, J., Sewell, H., & Watson, A. (2004). *Primary questions and prompts*. Association of Teachers of Mathematics (ATM).

K12 Computer Science Framework. (2016). *K12 Computer Science Framework*. https://k12cs.org

Kallia, M., van Borkulo, S., Drijvers, P., Barendsen, E., & Tolboom, J. (2021). Characterising computational thinking in mathematics education: A literature-informed Delphi-study. *Research in Mathematics Education*, *23*(2), 159–187. https://doi.org/10.1080/14794802.2020.1852104

Kelter, J., Peel, A., Bain, C., Anton, G., Dabholkar, S., Horn, M. S., & Wilensky, U. (2021). Constructionist co-design: A dual approach to curriculum and professional development. *British Journal of Educational Technology*, *52*(3), 1043–1059. https://doi.org/10.1111/bjet.13084

Knuth, D. E. (1985). Algorithmic thinking and mathematical thinking. *The American Mathematical Monthly*, *92*(1), 170–181. https://doi.org/10.1080/00029890.1985.11971572

Lagrange, J. B., & Rogalski, J. (2017). Savoirs, concepts et situations dans les premiers apprentissages en programmation et en algorithmique [Knowledge, concepts and situations in early learning in programming and algorithms]. *Annales de Didactiques et de Sciences Cognitives*, *22*, 119–158. https://hal.archives-ouvertes.fr/hal-01740442/document

Lee, I. (2016). Reclaiming the roots of CT. *CSTA Voice*, *12*(1), 3–4.

Let's Talk Science. (2018). *Computational thinking framework*. https://letstalkscience.ca/sites/default/files/2019-10/LTS-Computational_Thinking_Framework-2018.pdf

Lockwood, E., Asay, A., DeJarnette, A. F., & Thomas, M. (2016). Algorithmic thinking: An initial characterization of computational thinking in mathematics. In M. B. Wood, E. E. Turner, M. Civil, & J. A. Eli (Eds.), *Proceedings of the 38th annual meeting of the North*

American Chapter of the International Group for the Psychology of Mathematics Education (pp. 1588–1595). The University of Arizona. https://files.eric.ed.gov/fulltext/ED583797.pdf

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

Lodi, M., & Martini, S. (2021). Computational thinking, between Papert and Wing. *Science & Education*, *30*(4), 883–908. https://doi.org/10.1007/s11191-021-00202-5

Meyer, A., & Modeste, S. (2018). Recherche binaire et méthode de dichotomie, comparaison et enjeux didactiques à l'interface mathématiques-informatique [Binary research and method of dichotomy, comparison and didactic issues at the mathematics-computer science interface]. In M. Abboud (Ed.), *Mathématiques en scènes, des ponts entre les disciplines: Actes du colloque EMF 2018* (pp. 1658-1667). IREM. https://publimath.univ-irem.fr/numerisation/ACF/ACF19269/ACF19269.pdf

Meyer, A., & Modeste, S. (2020). Analyse didactique d'un jeu de recherche: Vers une situation fondamentale pour la complexité d'algorithmes et de problèmes [Didactic analysis of a research game: Towards a fundamental situation for the complexity of algorithms and problems]. In P.-A. Caron, C Fluckiger, P. Marquet, Y. Peter, & Y. Secq (Eds.), *L'informatique, objets d'enseignements enjeux épistémologiques, didactiques et de formation: Actes du colloque DIDAPRO 8* (pp. 136–147). Université de Lille. https://hal.archives-ouvertes.fr/hal-02474983/document

Ministère de l'éducation nationale, de l'enseignement supérieur et de la recherche. (2015). *Programmes pour les cycles 2, 3, 4.* http://cache.media.education.gouv.fr/file/MEN_SPE_11/67/3/2015_programmes_cycles234_4_12_ok_508673.pdf

Ministère de l'éducation nationale, de l'enseignement supérieur et de la recherche. (2016). *Initiation à la programmation aux cycles 2 et 3*. https://eduscol.education.fr/document/15409/download

Ministère de l'éducation nationale, de l'enseignement supérieur et de la recherche. (2019). Programme d'enseignement de spécialité de mathématiques de la classe de première de la voie générale. *BO spécial n°1 du 22 janvier.* https://www.education.gouv.fr/bo/19/Special1/MENE1901632A.htm

Modeste S. (2012a). Enseigner l'algorithme pour quoi? Quelles nouvelles questions pour les mathématiques? Quels apports pour l'apprentissage de la preuve? [Teach the algorithm for what? What new questions for mathematics? What contributions for learning the proof?] [Doctoral dissertation, Université de Grenoble]. https://tel.archives-ouvertes.fr/tel-00783294

Modeste, S. (2012b). La pensée algorithmique: Apports d'un point de vue extérieur aux mathématiques [Algorithmic thinking: Contributions from a point of view outside mathematics]. In J.-L. Dorier & S. Coutat (Eds.), *Enseignement des mathématiques et contrat social: Enjeux et défis pour le 21e siècle: Actes du Colloque EMF 2012 (GT3, pp. 467–479).* http://www.emf2012.unige.ch/index.php/actes-emf-201

Modeste S. (2015). Impact of informatics on mathematics and its teaching. On the importance of epistemological analysis to feed didactical research. In F. Gadducci & M. Tavosanis

(Eds.), *History and philosophy of computing* (pp. 243–255). Springer. https://doi.org/10.1007/978-3-319-47286-7_17

Modeste S., & Ouvrier-Buffet C. (2011). The appearance of algorithms in curricula, a new opportunity to deal with proof? In M. Pytlak, T. Rowland, & E. Swoboda (Eds.), *Proceedings of CERME7: Seventh Congress of the European Society for Research in Mathematics Education* (pp. 202–212). https://hal.archives-ouvertes.fr/hal-03184702

New Zealand Ministry of Education. (2022). *Digital technologies and Hangarau Matihiko learning*. https://www.education.govt.nz/our-work/changes-in-education/digital-technologies-and-hangarau-matihiko-learning/

Nijimbere, C. (2015). *L'enseignement de savoirs informatiques pour débutants, du second cycle de la scolarité secondaire scientifique à l'université en France: Une étude comparative* [The teaching of computer knowledge for beginners, from the second cycle of scientific secondary schooling to university in France: A comparative study] [Doctoral dissertation, Université Paris Descartes]. Agence bibliographie de l'enseignement supérieur. http://www.theses.fr/2015USPCB086#

Noss, R., & Hoyles, C. (1996). *Windows on mathematical meanings: Learning cultures and computers* (Vol. 17). Kluwer. https://doi.org/10.1007/978-94-009-1696-8

OECD. (2018). *PISA 2021 mathematics framework (draft)*. https://www.oecd.org/pisa/sitedocument/PISA-2021-mathematics-framework.pdf

Ouvrier-Buffet, C., Meyer, A., Modeste, S. (2018). Discrete mathematics at university level. Interfacing mathematics, computer science and arithmetic. In V. Durand-Guerrier, R. Hochmuth, S. Goodchild, & N. M. Hogstad (Eds.), *Proceedings of INDRUM 2018: Second Conference of the International Network for Didactic Research in University Mathematics* (pp. 255–264). University of Agder and INDRUM. https://doi.org/10.1007/978-3-319-77487-9_51-5

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. https://doi.org/10.1080/0020739700030306

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

Papert, S., & Harel, I. (1991). Situating constructionism. In I. Harel & S. Papert (Eds.), *Constructionism* (pp. 1–11). Ablex.

Passey, D. (2017). Computer science (CS) in the compulsory education curriculum: Implications for future research. *Education and Information Technologies*, *22*(2), 421–443. https://doi.org/10.1007/s10639-016-9475-z

Perrin-Glorian, M.-J. (2011). L'ingénierie didactique à l'interface de la recherche avec l'enseignement: Développement de ressources et formation des enseignants [Didactic engineering at the interface of research and teaching: Resource development and teacher training]. In C. Margolinas, M. Abboud-Blanchard, L. Bueno-Ravel, N. Douek, A. Fluckiger, P. Gibel, F. Vanderbrouck, & F. Wozniak (Eds.), *En amont et en aval des ingénieries didactiques* (pp. 55-76). La Pensée Sauvage.

Pólya, G. (1970). *How to solve it*. Doubleday.

Rafalska, M. (2019). Towards improving teaching and learning of algorithmics by means of resources design: A case of primary school education in France. In U. T. Jankvist, M. van den Heuvel-Panhuizen, & M. Veldhuis (Eds.), *Proceedings of the Eleventh Congress of the European Society for Research in Mathematics Education (CERME11)* (pp.

4302-4309). Freudenthal Group & Freudenthal Institute, Utrecht University and ERME. https://hal.archives-ouvertes.fr/hal-02423497/

Rafalska, M. (2022). Task design for promoting pupils' algorithmic thinking in problem-solving context without using computers *In Proceedings of the Twelfth Congress of the European Society for Research in Mathematics Education. European Society for Research in Mathematics Education (ERME).* https://hal.science/CERME12/hal-03748490v1

Reichert, J. T., Barone, D. A. C., & Kist, M. (2020). Computational thinking in K-12: An analysis with mathematics teachers. *EURASIA Journal of Mathematics, Science and Technology Education*, *16*(6), Article em1847. https://doi.org/10.29333/ejmste/7832

Resnick, M. (2014). Give P's a chance: Projects, peers, passion, play. In G. Futschek & C. Kynigos (Eds.), *Constructionism and creativity: Proceedings of the Third International Constructionism Conference* (pp. 13-20). Austrian Computer Society.

Rich, K. M., Yadav, A., & Larimore, R. A. (2020). Teacher implementation profiles for integrating computational thinking into elementary mathematics and science instruction. *Education and Information Technologies*, 25, 3161–3188. https://doi.org/10.1007/s10639-020-10115-5

Rich, K. M., Yadav, A., & Schwarz, C. V. (2019). Computational thinking, mathematics, and science: Elementary teachers' perspectives on integration. *Journal of Technology and Teacher Education*, *27*(2), 165–205. https://par.nsf.gov/servlets/purl/10183080

Sands, P., Yadav, A., & Good, J. (2018). Computational thinking in K–12: In-service teacher perceptions of computational thinking. In M. S. Khine (Ed.), *Computational thinking in the STEM disciplines* (pp. 151–164). Springer. https://doi.org/10.1007/978-3-319-93566-9_8

Sangwin, C. J., & O'Toole, C. (2017). Computer programming in the UK undergraduate mathematics curriculum. *International Journal of Mathematical Education in Science and Technology*, *48*(8), 1133–1152. https://doi.org/10.1080/0020739X.2017.1315186

Shulman, L. S. (1986). Those who understand: Knowledge growth in teaching. *Educational Researcher*, *15*(2), 4–14. https://doi.org/10.3102/0013189X015002004

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

So, H.-J., Jong, M. S.-Y., & Liu, C.-C. (2020). Computational thinking education in the Asian pacific region. *The Asia-Pacific Education Researcher*, *29*(1), 1–8. https://doi.org/10.1007/s40299-019-00494-w

Suters, L. (2021). Elementary preservice teacher coursework design for developing science and mathematics computational thinking practices. *Contemporary Issues in Technology and Teacher Education*, *21*(2), 360–440.

Tempier, F. (2016). New perspectives for didactical engineering. An example for the development of a resource for teaching decimal number system. *Journal of Mathematics Teacher Education*, *19*, 261–276. https://doi.org/10.1007/s10857-015-9333-8

Waite, J., & Grover, S. (2020). Worked examples and other scaffolding strategies. In S. Grover (Ed.), *Computer science in K–12: An A-to-Z handbook on teaching programming* (pp. 240-249). Edfinity.

Webb, M., Davis, N., Bell, T., Katz, Y. J., Reynolds, N., Chambers, D. P., & Syslo, M. M. (2017). Computer science in K-12 school curricula of the 21st century: Why, what and

when? *Education and Information Technologies*, *22*, 445–468. https://doi.org/10.1007/s10639-016-9493-x

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

Wiebe, E., Kite, V., & Park, S. (2020). Integrating computational thinking in STEM. In C. C. Johnson, M. J. Mohr-Schroeder, T. J. Moore, & L. D. English (Eds.), *Handbook of research on STEM education* (pp. 196–209). Routledge.

Wilensky, U. (1995). Paradox, programming and learning probability. *Journal of Mathematical Behavior*, *14*(2), 253–280. https://doi.org/10.1016/0732-3123(95)90010-1

Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, *49*(3), 33–36. https://doi.org/10.1145/1118178.1118215

Wing, J. M. (2010). *Computational thinking: What and why?* [Unpublished manuscript]. Computer Science Department, Carnegie Mellon University. https://www.cs.cmu.edu/~CompThink/resources/TheLinkWing.pdf

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