Kampylis, P., Dagienė, V., Bocconi, S., Chioccariello, A., Engelhardt, K., Stupurienė, G., Masiulionytė-Dagienė, V., Jasutė, E., Malagoli, C., Horvath, M., & Earp, J. (2023). Integrating Computational Thinking into Primary and Lower Secondary Education: A Systematic Review. *Educational Technology & Society*, *26*(2), 99-117. https://doi.org/10.30191/ETS.202304_26(2).0008

Integrating Computational Thinking into Primary and Lower Secondary Education: A Systematic Review

Panagiotis Kampylis[1], Valentina Dagienė[2], Stefania Bocconi[1*], Augusto Chioccariello[1], Katja Engelhardt[3], Gabrielė Stupurienė[2], Vaida Masiulionytė-Dagienė[2], Eglė Jasutė[2], Chiara Malagoli[1], Milena Horvath[3] and Jeffrey Earp[1]

[1]National Research Council, Italy // [2]Vilnius University, Lithuania // [3]European Schoolnet, Belgium // panagiotis.kampylis@itd.cnr.it // valentina.dagiene@mif.vu.lt // stefania.bocconi@itd.cnr.it // augusto@itd.cnr.it // katja.engelhardt@outlook.com // gabriele.stupuriene@mif.vu.lt // vaida.masiulionyte-dagiene@mif.vu.lt // egle.jasute@fsf.vu.lt // chiara.malagoli@itd.cnr.it // milena.horvath@eun.org // jeffrey.earp@itd.cnr.it

[*]Corresponding author

ABSTRACT: In recent years, many countries have introduced Computational Thinking (CT) concepts into compulsory education as part of general curriculum reform efforts. A systematic review of academic and grey literature has been conducted to analyse the state of the art in implementing CT in primary and secondary education. In total, 1977 publications were identified, out of which 98 met the inclusion criteria for the review. The results show that, despite a lack of consensus on a common definition, a core set of key CT skills is addressed in primary and lower secondary education. Implementation approaches that emerged from the analysis are discussed and presented according to the European Commission's Joint Research Centre (2016) classification: (i) embedding CT across the curriculum as a transversal theme/skill set; (ii) integrating CT as a separate subject; and (iii) incorporating CT skills within other subjects such as Mathematics and Technology. New approaches to formative assessment of CT are emerging, reflecting different conceptualisations and differences in contextual and motivational aspects of CT curriculum integration. However, further investigation is needed to understand better how gender/equity/inclusion issues impact the quality of computing education integration.

Keywords: Computational thinking, Computer Science education, Compulsory education, CT skills

1. Introduction

The Digital Education Action Plan 2021-2027 (European Commission, 2020b, p. 94) provides the following glossary definition of CT (along with programming and coding):

> Computational thinking, programming and coding are often used in an interchangeable way in education settings, but they are distinct activities. Programming refers to the activity of analysing a problem, designing a solution and implementing it. Coding means implementing solutions in a particular programming language. Computational thinking, shorthand for "thinking as a computer scientist," refers to the ability to understand the underlying notions and mechanisms of digital technologies to formulate and solve problems.

This conceptualisation reflects the definition proposed by Wing (2017, p. 8): "Computational thinking is the thought processes involved in formulating a problem and expressing its solution(s) in such a way that a computer — human or machine — can effectively carry out." This conceptualisation is clearly connected to the concepts and practices of Computer Sciences - CS (i.e., thinking like a computer scientist) proposed as an intellectual framework for thinking. Caeli and Yadav (2020) provided the historical perspectives of CT and how initiatives today can inspire students to learn CS. A deeper historical development of CT and the intellectual ideas for development are provided by Tedre and Denning (2016).

In this paper, we use the term *Computer Science* (CS) interchangeably with *Computing* and *Informatics*. In Wing's view (2017, p. 7), "computational thinking will be a fundamental skill - just like reading, writing, and arithmetic - used by everyone by the middle of the 21st Century." This vision has been widely accepted as a basis for including CT as a key 21st century competence in compulsory education (e.g., Pérez-Marín et al., 2020). However, discussions on understanding of CT and its importance in schools have continued. As Curzon et al. (2019) note, this debate can be represented as positioning CT applicability on a scale that goes from the broad (e.g., the CT skillset overlaps with skills in other disciplines and is generally useful; computational systems exist in the natural world) to the narrow (e.g., CT is not necessarily beneficial for everyone; computational systems are confined to computers). However, most CT definitions in the literature position it somewhere between these two extremes (e.g., CT skills include algorithmic thinking, logical thinking, abstraction, generalisation, and decomposition; CT is also relevant to non-technical disciplines). Further support for this interpretation comes from the OECD’s PISA and IEA’s ICILS educational assessments, which explicitly refer to CT and include key CT skills and concepts in their tests (e.g., Fraillon et al., 2019).

Generally, CT is regarded in the literature as a thought process involved in designing solutions that a computer, a human, or both can execute. While numerous definitions are currently being proposed and adopted, an agreement is nonetheless emerging on core CT concepts, namely abstraction, algorithmic thinking, automation, decomposition, and generalisation (Curzon et al., 2019). These are closely connected with a set of attitudes and skills (named as practices) that comprises creating computational artefacts, testing and debugging, collaboration and creativity, and the capacity to tackle open-ended problems (Grover & Pea, 2018). In this view, CT is seen as a fundamental competence for informed citizens to manage the ever-emerging challenges society poses. In addition, CT offers the potential to support creative problem-solving and may foster innovative approaches in other subject areas. Hence, it undoubtedly has a key role to play in compulsory education. By making CT concepts concrete, programming provides opportunities for CT education. This renders it a tool for learning, e.g., a way of exploring other domains or a means for self-expression (Resnick, 2017). Nevertheless, it is generally agreed that coding/programming is just one of the various facets of CT.

This paper is intended to provide an overview of the integration of CT skills in primary and lower secondary education, as emerging from recent research works. The systematic literature review presented herein builds upon and complements that presented by Bocconi et al. (2016), capturing developments in the CT field from 2016 to 2021. We use the term *Computer Science* (CS) for both Computing and Informatics, in line with the Digital Education Action Plan 2021-2017 (European Commission, 2020a, p. 13).

To understand how the ongoing debate and results from research on CT are influencing the implementation of CT-related curricula in schools, the following research questions were addressed:

- RQ1: How is CT defined in the context of compulsory education?
- RQ2: How is CT implemented in primary and lower secondary education?
- RQ3: How are gender and equity addressed when implementing CT in the curriculum?

Section 2 presents the methodology adopted for conducting the systematic literature review. In Section 3, we present and discuss the review results, with particular attention devoted to CT definitions in various settings, the curricular issues intertwined with CS education, gender balance and equity, pedagogical approaches adopted, and technologies employed. Some conclusions are provided in the final section.

2. Method

2.1. The PRISMA 2020 statement

A structured approach was employed to identify relevant academic and grey literature and select the publications to be analysed in-depth. The Preferred Reporting Items for Systematic Reviews and Meta-Analyses – PRISMA 2020 statement (Page et al., 2021) was applied to increase the dependability and reliability of the data collected and analysed. Figure 1 presents the PRISMA 2020 workflow steps followed (Identification, Screening, Included) and the number of records handled in each step.

2.2. Identification process

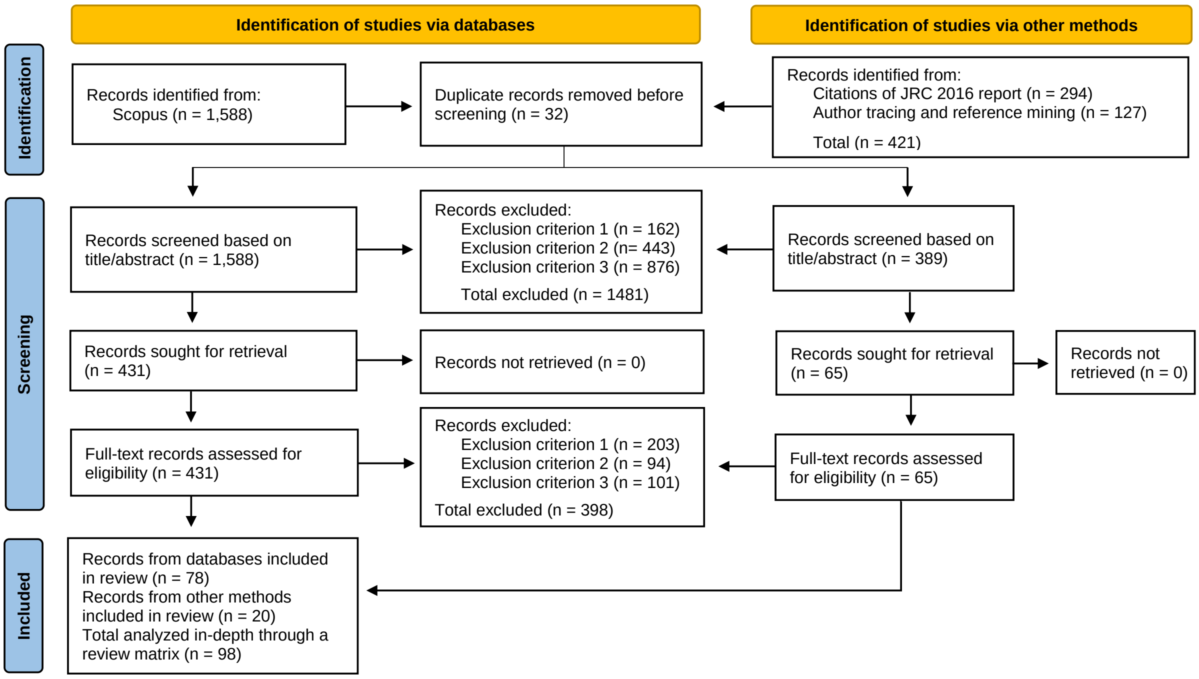
The initial search was conducted on the Scopus research database in May 2021, focusing on a selection of 24 top-tier journals and conference proceedings (Table 1) devoted to pedagogical aspects of CT and CS education.

A broad coverage of studies was sought in searching for CT occurrences and related terms. Accordingly, the following Boolean string was used to identify relevant papers published after 2016 in the selected journals: (ISSN (XXXX-XXXX) AND ALL (“computational thinking”) OR ALL (“algorithmic thinking”) OR ALL (“computer science education”) OR ALL (“computing education”) OR ALL (“informatics education”) AND PUBYEAR > 2015). For the conference proceedings, the following search string was employed: (ALL (“computational thinking”) OR ALL (“algorithmic thinking”) AND CONF (XXXXX) AND PUBYEAR > 2015). The search string for identifying conference papers did not include the search terms “computer science education,” “computing education,” or “informatics education” because the selected conference proceedings are specialised in these fields, in contrast with the selected journals, which are wider in their scope.

Table 1. Number of publications identified, collected, and screened per source

Type	Source	Identified and screened		
		Step 1	Step 2	Step 3
Journals	Education and Information Technologies	158	27	2
	Computers & Education	135	26	13
	ACM Transactions on Computing Education	133	25	2
	Computer Science Education	111	22	3
	Computers in Human Behavior	87	19	3
	Journal of Educational Computing Research	80	15	5
	Informatics in Education	63	14	8
	IEEE Transactions on Education	61	7	0
	International Journal of Child-Computer Interaction	53	16	0
	Journal of Computer Assisted Learning	45	11	2
	TechTrends	36	7	4
	Journal of Research on Technology in Education	21	10	2
	Thinking Skills and Creativity	12	1	0
	Comunicar	7	1	0
	Journal papers citing CompuThink 2016 study	122	23	2
	Journal papers from author tracing and reference mining	2	2	2
Edited books	Books chapters citing CompuThink 2016 study	17	4	2
	Book chapters from author tracing and reference mining	3	3	3
Conference proceedings	SIGCSE: Symposium on CS Education	327	120	11
	ITiCSE: Innovation and Technology in CS Education	80	28	0
	ISSEP: Informatics in School Education: Evolution and Perspectives	63	29	6
	WIPSCE: Workshop in Primary and Secondary Computing Education	60	33	2
	Koli Calling Conference on Computing Education Research	48	15	4
	LaTiCE: Learning and Teaching in Computing and Engineering	8	5	0
	Conference papers citing Computhink 2016 study	50	8	0
	Conference papers from author tracing and reference mining	2	2	2
Grey literature	Grey literature citing ComputThink 2016 study	73	18	7
	Grey literature from author tracing and reference mining	120	25	13
	Total	1977	516	98

Figure 1. PRISMA 2020 Flow Diagram for systematic literature review. Adapted from Page et al. (2021)


Identification of studies via databases
Identification of studies via other methods
Identification
Screening
Included
Records identified from: Scopus (n = 1,588)
Duplicate records removed before screening (n = 32)
Records identified from: Citations of JRC 2016 report (n = 294) Author tracing and reference mining (n = 127) Total (n = 421)
Records screened based on title/abstract (n = 1,588)
Records excluded: Exclusion criterion 1 (n = 162) Exclusion criterion 2 (n= 443) Exclusion criterion 3 (n = 876) Total excluded (n = 1481)
Records screened based on title/abstract (n = 389)
Records sought for retrieval (n = 431)
Records not retrieved (n = 0)
Records sought for retrieval (n = 65)
Records not retrieved (n = 0)
Full-text records assessed for eligibility (n = 431)
Records excluded: Exclusion criterion 1 (n = 203) Exclusion criterion 2 (n = 94) Exclusion criterion 3 (n = 101) Total excluded (n = 398)
Full-text records assessed for eligibility (n = 65)
Records from databases included in review (n = 78) Records from other methods included in review (n = 20) Total analyzed in-depth through a review matrix (n = 98)

In addition to the Scopus search, we employed citation tracking to identify post-2016 academic and grey literature, which has become pivotal in CT research and computing education. These works include conceptually oriented and empirical studies that have (a) generated a line of investigation which has changed how problems or questions have been framed, (b) introduced new methods or concepts, or (c) generated influential debate. We also employed Google Scholar to identify and gather publications that cite the systematic literature review by Bocconi et al. (2016), referred to as the *CompuThink 2016* study.

2.3. Screening and eligibility process

The screening process was performed in three steps. In Step 1, the 1977 aggregated publications were allocated among the nine researchers involved in the literature review. Screening of records involved reading the title and abstract and applying three exclusion criteria: (i) not a full article; (ii) not devoted to compulsory education; (iii) devoted to specialised topics (e.g., cybersecurity, machine learning, data analytics) outside the scope of the study.

In Step 2, the full texts of the 516 records potentially eligible for in-depth analysis were collected and subsequently screened by applying the following exclusion criteria: (i) tangential or no specific focus at all on CT/CS; (ii) pilot studies of low quality and/or conducted on small sample size; (iii) empirical papers reporting outcomes not explicitly concerning CT/CS.

In Step 3, the 98 publications (see Table 1) from academic (N = 78) and grey literature (N = 20) were distributed among the nine researchers and analysed in-depth through a review matrix (see Appendix 1) following specific guidelines and a shared understanding of the different fields, as described in the following section.

2.4. Quality assurance

To ensure that the researchers analysed and coded the selected literature homogeneously and comparably, Inter-Rater Reliability (IRR) checks were carried out as a measure of quality assurance. Furthermore, guidelines were drafted for researchers to follow throughout the PRISMA screening process and for mapping against the review matrix.

To precisely evaluate IRR for multiple non-unique coders, an extension of the k statistic was used, as propounded by Hallgren (2012). Cohen’s kappa was calculated through SPSS version 26.0, and the arithmetic mean of these outputs was computed. This process determined the level of agreement among all the researchers on whether 24 publications of different nature, including academic and grey literature, could be included or excluded from the literature review based on guidelines for the PRISMA screening process.

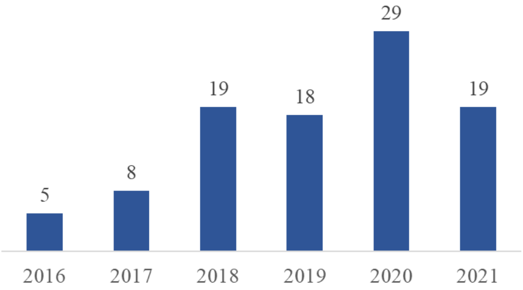
To avoid influencing the other researchers’ decisions, each researcher rated the same set of 24 randomly selected publications individually. When evaluating publications, each researcher could choose only to include or exclude them for screening in Step 3. Researchers were asked to perform two rounds of evaluation. Overall, in round one, a moderate level of coder agreement was reached (Mean κ = .41). In line with the approach proposed by Belur et al. (2018), this first round of coding was followed by an open discussion among members of the coding team led by two senior researchers/coders. This discussion permitted clarification of the inclusion/exclusion criteria, resulting in an excellent level of agreement in the second round (Mean κ = .98).

3. Results and discussion

Following the process described in the previous section, the 98 publications from 2016 to 2021 were analysed in depth. In terms of temporal spread, a significant increase in 2020 emerges (see Figure 2).

A thorough analysis of the collected works was carried out through a review matrix approach (see Appendix 1), thus facilitating a structured comparison of different sources. The research questions in the matrix are broader than those addressed in this paper, as the matrix was designed for comprehensive extraction and documentation of all the insights from each publication, thereby providing a solid basis for systematic analysis.

Figure 2. Distribution of analysed literature by year of publication



3.1. CT definition and concepts in different settings

Although this field has been the subject of intensive research for about 15 years, there is still debate on the definition of CT. Different research teams have tended to expound their own CT definition informed by their specific line of inquiry and have assumed different perspectives regarding applying, interpreting, and assessing proposed CT concepts. Taslibeyaz et al. (2020) note that CT definitions are often context-specific.

Tikva and Tambouris (2021) categorise CT definitions as domain-specific or domain-general. Domain-specific definitions indicate domain-specific knowledge or skills needed to solve problems systematically in the subject area of CS or programming. Domain-general definitions refer to competences necessary for solving problems systematically in all learning activities. This framework is similar to that proposed by Tang et al. (2020), which divides CT definitions related to (i) programming and computing concepts, and (ii) competences.

The three-type categorisation of CT definitions (i.e., generic, operational and educational / curricular definitions) proposed by Román-González et al. (2017) is used below to present examples of CT definitions emerging from the analysed literature (see Table 2).

Those who offer a precise definition of CT agree that it is a mode of thinking (thought process) for problem-solving (Grover & Pea, 2018; Hazzan et al., 2020; Zhang & Nouri, 2019). However, whatever one's view on the definition of CT, it is important to be pragmatic regarding the best ways of teaching it (Curzon et al., 2019). What all definitions have in common is that CT is more than problem-solving: the problem's solution must be expressed in such a way that permits it to be executed by a computational agent. CT is the way of thinking for developing solutions that allow a processing agent (machine) to carry it out (Corradini et al., 2017; Curzon et al., 2019; Csizmadia et al., 2019).

According to Fessakis and Prantsoudi (2019), CT-related skills commonly cited in various definitions are: algorithmic approach to problem-solving (including creativity), abstraction, logical reasoning, problem-solution transfer, generalisation, processing of data, and social impact of computation. In addition, Csizmadia et al. (2019) suggest the combination of CT with constructionism for selecting and evaluating classroom activities.

Corradini at al. (2018) classified all constitutive elements of CT into four categories: (i) mental processes or strategies useful to solve problems; (ii) methods, i.e., operational approaches used by computer scientists; (iii) practices used in the implementation of computer-based solutions; and (iv) transversal skills, e.g., general skills enhanced by CS application.

In a systematic review of empirical studies (Tang et al., 2020), the authors analyse well-cited CT definitions and notice that many are related to programming and computing. Tikva and Tambouris (2021, p. 162) observe a reciprocal association between CT and programming: "programming supports the development of CT while CT provides to programming a new upgraded role." However, Hazzan et al. (2020, p. 61) summarise that "CT is not necessarily about programming, but rather, the emphasis is on problem-solving," which fosters learning experiences. Nevertheless, programming is still the most frequently mentioned concept taught and, as Upadhyaya et al. (2020) remark, programming coupled with abstraction is becoming more commonly mentioned in conjunction with CT skills. CT is often conceptualised in a programming context and can be examined in terms of three key components: CT concepts, CT practices, and CT perspectives (Kong et al., 2020).

Webb et al. (2017, p. 449) state:

> The distinction between computational thinking and programming is subtle; in principle computational thinking does not require programming at all, although in practice, representing a solution to a problem as a

program provides a perfect way to evaluate the solution, as the computer will execute the instructions to the letter, forcing the student to refine their solution so that it is very precise.

Table 2. The three types of CT definitions with examples

CT definition categories	Examples of CT definitions in the analysed literature
Generic definition	CT is the thought process entailed in formulating a problem and expressing the solution(s) so that a computer-human or machine can perform it effectively (Grover & Pea, 2018; Rich et al., 2021). CT regards thinking processes, so its implementation is independent of technology (Hazzan et al., 2020). CT is a thought process involving fundamental programming skills (CT skills) for solving problems in any domain (Zhang & Nouri, 2019).
Operational or model definition	The CT framework employs fundamental CS concepts to solve problems, design systems, and understand human behaviour (Jocius et al., 2020). CT encompasses a set of broadly applicable problem-solving skills that include abstraction, algorithmic thinking, decomposition, and pattern recognition (Huang & Looi, 2020). CT is a means to understand and solve complex problems by using CS concepts and techniques such as decomposition, pattern recognition, abstraction, and algorithms (Kale et al., 2018). Eight aspects at CT's core are highlighted: abstraction, algorithm design, evaluation, generalisation, iterative improvement, information representation, precise communication, and problem decomposition (Komm et al., 2020). CT definition relates to the operationalisation of CT practices (mainly based on Zoombinis gameplay) and focuses on four CT practices: problem decomposition, pattern recognition, abstraction, and algorithm design (Asbell-Clarke et al., 2021). CT is defined as a conceptual framework based on the five fundamental CT concepts: abstraction, decomposition, algorithmic thinking, evaluation, and generalisation (Tsai et al., 2020). The computing-based CT definition framework divided CT into general practices such as data, modelling & simulation, computational problem-solving, and systems thinking (Weintrop & Wilensky, 2019).
Educational and curricular definitions	CT entails addressing problem-solving systematically (e.g., algorithmically) so that the solutions generated can be reused in various contexts (Shute et al., 2017). According to the Australian Curriculum, Assessment and Reporting Authority, CT is a problem-solving approach entailing various strategies and techniques that can be implemented using computer systems (Australian Computing Academy, 2019). CT is considered a means to develop knowledge and understand concepts in CS and contributes significantly to general problem-solving skills (Israel-Fishelson & Hershkovitz, 2020). CT encompasses four different computational practices (problem-solving or algorithmic thinking, building algorithms, debugging, and simulation) and some concepts (Hooshyar et al., 2020). Thinking computationally means employing CS principles and methods to efficiently address and solve problems (Arfé et al., 2020) and developing algorithmic solutions to those problems so they can be operationalised using computers (Eickelmann et al., 2019).

Besides the theoretical discussion on CT definition, many studies investigate CT integration in classrooms. An operational definition of CT skills (see examples in Table 2) is more suitable for everyday activities and is broadly adopted in many studies (e.g., Barendsen et al., 2016; Grgurina et al., 2018; Leonard et al., 2021). Also, it is important to mention the historical perspectives of CT, which have a strong connection to computing (Caeli & Yadav, 2020; Tedre & Denning, 2016).

3.2. CT in the primary school curriculum

In a world where computing is pervasive, "CT is being recognised as a foundational competency for being an informed citizen and being successful in all STEM work, and potential for creative problem solving and innovating in all other disciplines" (Grover & Pea, 2018, p. 34).

Most researchers agree that programming should best be taught from a young age (Ching et al., 2018; Niemelä et al., 2017; del Olmo-Muñoz et al., 2020; Sáez-López et al., 2016; Wei et al., 2021). Usually, primary students are initially introduced to programming via unplugged activities, i.e., working without a computer or another digital device (del Olmo-Muñoz et al., 2020; Tonbuloğlu & Tonbuloğlu, 2019) and then move on to the use of block-based programming languages with computers (Arfé et al., 2020; Sáez-López et al., 2016; Sherwood et al., 2021). Related investigations have yielded solid evidence of the positive potential and results of developing CT when young learners program with different educational technologies (Ching et al., 2018; Kong et al., 2020). González-González et al. (2019) presented a study carried out with children aged of 3 to 6 years old with Down syndrome and showed that these pupils with cognitive disabilities can acquire basic programming and CT skills using tangible robots such as KIBO. Israel et al. (2020) examine how elementary students with autism behaved during computing instruction and concluded that these students require individualised support. Wei et al. (2021) report that partial pair programming effectively impacts the development of CT skills and self-efficacy in primary school students. Wu and Su (2021) have noticed that learning through physical robots can help students improve their CT abilities.

The primary education studies among the selected papers deal with three aspects: (i) programming in CS; (ii) programming in other disciplines like science, mathematics, art or integrated subjects; and (iii) programming used as a tool to assess CT concepts and skills. Generally, programming skills are developed in CS classes rather than in science or integrated disciplines. Based on the theoretical framework of programming-related CT proposed by Brennan and Resnick (2012), we investigated the selected papers and extracted examples of computational concepts, practices, and perspectives for primary education (see Table 3).

Table 3. Examples of CT-related concepts, practices, and perspectives in primary education

Key CT dimensions	Examples and references
Computational concepts	Sequences, directions (forward, back, left, right) and loops (del Olmo-Muñoz et al., 2020) Algorithms, automation, coordination, creativity, data, logic, modelling and design, patterns, and problem decomposition (Fagerlund et al., 2020) Algorithmic thinking, creativity, collaboration, critical thinking, and problem-solving (Tonbuloğlu & Tonbuloğlu, 2019) Parameters, passing parameters to subprograms, sequences, simple loops, repeat, variables (Hromkovič & Lacher, 2017) Data, sequences, loops, parallelism (Israel-Fishelson & Hershkovitz, 2020) Abstraction and connected concepts (Liebe & Camp, 2019; Statter & Armoni, 2020) The seven big ideas: creativity, abstraction, data, algorithms, programming, internet, and global impact (Repenning et al., 2021)
Computational practices	Abstracting and modularising, algorithmic thinking, data, computational practices, experimenting and iterating tests, reusing and remixing, testing and debugging (Basu et al., 2020; Román-González et al., 2017) Abstracting, algorithm design, pattern recognition, and problem decomposition (Asbell-Clarke et al., 2021; Rijke et al., 2018; Wu & Su, 2021) Algorithmic thinking, building algorithms, and debugging (Jocius et al., 2020) Coding, conditionals and testing, looping, functions or debugging, nested looping, and sequencing (Arfé et al., 2020; Israel-Fishelson et al., 2021) Data representation, flow control, parallelisation, and user interactivity (Wei et al., 2021)
Computational perspectives	Better understanding of the initialisation of variables and objects (Franklin et al., 2016)

Games are commonly used in teaching and learning in primary schools. Several studies examined the relationship between student gameplay and the development of students' CT practices using different classroom activities (Asbell-Clarke et al., 2021; Ching et al., 2018; Hooshyar et al., 2020; Israel-Fishelson & Hershkovitz, 2020).

To reinforce CT skills in the primary grades, unplugged activities in combination with constructionist approach to learning-to-think computationally are particularly helpful (Caeli & Yadav, 2020). Also, Csizmadia et al. (2019) developed and presented a new mapping tool (the constructionism matrix) to review classroom activities in terms of both CT and constructionist learning. Pérez-Marín et al. (2020) suggested a metaphor-based methodology using Scratch to teach primary school children basic algorithmic and programming concepts. They found a statistically significant increase in participants' results and concluded that it is possible to teach children basic computer programming concepts (conditionals, loops) and improve their CT skills.

3.3. CT in the lower secondary school curriculum

While in primary school CT skills are usually integrated into other subjects in primary school rather than being embedded in a specific dedicated subject, CT plays a much more complex role in secondary education. From their analysis of educational policy initiatives across the globe, Hsu et al. (2019) describe trends and rationales for including CT in the curricula. They highlight four main approaches: (i) creating a new subject; (ii) incorporating CT skills into existing subjects such as maths; (iii) embedding CT across the entire curriculum as a transversal skill set; (iv) combining the above. In many countries, introducing a new subject or even content area represents a serious challenge, as it usually entails making space in densely packed curricula. Rich, Mason and O'Leary (2021) developed the BootUp curriculum by gradually introducing coding and CT concepts and practices that are based on the K-12 CS Framework.

Jocius et al. (2020, p. 6) point out that "the value of computational thinking is not just as an isolated concept that relates to Eickelmann et al. (2019), but also as a way to enhance and support more complex discipline-specific and interdisciplinary understandings."

There is a strong emphasis on programming in CT skills development in compulsory education, especially in the secondary curriculum. Many researchers argue that programming improves understanding of CT concepts and contributes to CT practice (del Olmo-Muñoz et al., 2020; Wu & Su, 2020). Sáez-López et al. (2016) show that active pedagogical methods employing visual programming languages yield significant benefits for aspects like learning programming concepts and developing logic and computational practices. Researchers point out that programming stimulates the development of students' CT skills (Djambong et al., 2018; Corradini et al., 2018) and can be successfully employed for teaching CT in both primary and secondary schools (Hsu et al., 2019; Israel-Fishelson et al., 2021; Yağcı, 2019). Many secondary school teachers use program design courses to foster students' CT skills (Pasternak, 2016; Sáez-López et al., 2016; Zhang et al., 2020). Coenraad et al. (2021) designed Scratch Encore, a curriculum that uses Scratch and follows the Use-Modify-Create pedagogical strategy to introduce secondary school students to CS concepts in a culturally responsive way.

An increasing number of literature reviews published in the last few years investigate the broad association of CT skills with programming (Ching et al., 2018; Djambong et al., 2018; Hsu et al., 2019; Sun et al., 2021b; Zhang & Nouri, 2019). Several of these reviews concentrate on particular facets of the CT domain, such as programming; others address a spectrum of topics (e.g., Hsu et al., 2018; Ching et al., 2018). Often, teachers are expected to develop students' CT skills through program design courses based on structured assessment because algorithms and programming are recognised as ways to improve those skills (Sáez-López et al., 2016; Tikva & Tambouris, 2021; Román-González et al., 2019).

Numerous ongoing initiatives seek to foster CT skills by providing learners with programming tools and resources intended to facilitate the integration of CT in schools and thereby respond to societal needs for 21st-century skills (Hsu et al., 2019; Passey, 2017). Grover et al. (2019) developed a suite of non-programming digital activities embedded in a curriculum before students engage in Scratch block-based programming they use constructivism as a pedagogic approach. Tikva and Tambouris (2021) developed a conceptual model of CT within programming for K-12 education that is based on a systematic literature review summarising 101 studies and identifying CT areas. This model adopts a challenging holistic approach, seeking to support CT teaching and learning in K-12 education. Dagli and Sancar Tokmak (2021) emphasise development of students' CT skills through instructional design stages: analysis, design, development, implementation, evaluation and revision.

Palts and Pedaste (2020) propose an innovative model for developing CT skills This is based on three stages in CT training: (i) define the problem, including formulation and reformulation, abstraction, and decomposition; (ii) solve the problem by using data analysis, algorithm design, parallelisation, iteration and automation; and (iii) analyse the solution based on generalisation, testing and evaluation.

Grover and Pea (2018, p. 34) state that:

> […] learning CT, much like learning scientific and mathematical thinking, is more about developing a set of problem-solving heuristics, approaches and 'habits of mind' than simply learning how to use a programming tool to create computational artefacts.

Li (2020) stresses that good design for developing CT skills includes several important aspects, such as scaffolding, learner-centred methods, and fostering deep learning by integrating CT skills in teaching coding.

3.4. Integrating CT as a separate subject

CT is increasingly becoming a central focus as countries update their school curricula. UNESCO and IFIP TC3 (2019) recommend promoting CT through the CS curriculum based on the understanding that CT is integral to problem-solving approaches in CS. Fessakis and Prantsoudi (2019) remind us that CT was introduced as a conceptual tool to promote the status of CS in general education. Accordingly, CT enables students to leverage advancements made possible by CS, from data collection to in-depth research. One important goal of today's CS curricula is improving students' CT skills (Kert et al., 2019). Tomokyio (2018) explored successes and challenges in implementing a progressive CS curriculum for K-8 schools. A new comprehensive curriculum that integrates various CS topics in middle schools in Qatar was elaborated and discussed by Razak et al. (2021). Forlizzi et al. (2018) proposed a core CS (Informatics) curriculum for all the levels of compulsory school, and outlined strategies to ensure that its implementation in schools can be effective.

The Australian Computing Academy (2019, p. 30) states that:

> The development of pedagogy in computer science education lags behind that of other subjects. In contrast to CS, mathematics has been taught at schools for centuries, and there is broad consensus about teaching key concepts at different year levels, taking into account the changing cognitive capabilities as students age.

The CS (Informatics) curriculum in the Netherlands can serve as an example of incorporating CT education (Barendsen et al., 2016). The core of the curriculum comprises a skill set (comprising not just CS-specific skills but also general scientific and technical skills), together with five knowledge domains. In addition, three skill subdomains are deemed crucial aspects characterising CS as a subject: (i) design and development; (ii) focus on informatics perspective; and (iii) collaboration and interdisciplinarity. A few years following the curriculum's introduction, Grgurina et al. (2018) developed a curriculum intervention including a practical assignment and an accompanying assessment instrument consisting of grading rubrics.

When applying a strategy for a CS curriculum integrated with CT, Hromkovič and Lacher (2017) espouse the following principles:

- CS must not be taught as an isolated subject but rather as a part of Science and Technology, thus providing an in-depth contextual view;
- Do not teach the use of the latest IT products or the latest scientific discoveries. Instead, investigate the evolution of fundamental concepts and their step-by-step development;
- Teach programming and automate well-understood activities with computers.

Hromkovič et al. (2016) envisioned a broad and comprehensive CS education from primary school by introducing Logo and then carefully building through secondary school using text-based programming languages such like Python.

Dealing with approaches for CT education, Li (2020, p. 10) emphasised that:

> […] experts suggest that although CT can certainly be taught through CS, it may not be the best approach for different reasons. First, CT [skills are] best learned when they are integrated into different subjects for elementary students. Secondly, even for secondary students, depending on the student group and school environments, integrating CT into different subjects may also prove to be the most practical and useful way.

Numerous recently published studies suggest that programming skills should be considered fundamental skills that are as important as reading and writing, so CT and programming are deeply intertwined (Metcalf et al., 2021). CT shifts the focus from programming and learning to code to areas like problem-solving in various disciplines using coding or other CT skills (Basu et al., 2020). Many researchers describe the connection between programming and CT in compulsory education in the context of a CS curriculum (Sun et al., 2021b; Tikva & Tambouris, 2021; Waite et al., 2020; Webb et al., 2017). CS itself ranges from the digital skills needed to *use* technology to advanced programming skills required to *design* that technology.

When discussing unplugged pedagogy as a way to support CS-for-all and CT development, Huang and Looi (2020) point to the plethora of unplugged activities that have been developed and adapted as proof that the approach offers flexibility and suitability for teaching a broad range of learners.

3.5. Integrating CT within other subjects

Given these practical considerations, integrating CT across subjects transversally offers considerable advantages (Balanskat et al., 2018). CT skills are not necessarily confined to development within a single subject but rather can be seen as a set of essential thinking skills applicable to any STEM-related field (Sun et al., 2021b).

Non-CS-focused approaches aim to integrate CT into learning experiences within subjects other than CS, such as STEM - Science, Technology, Engineering, and Mathematics (Li, 2020; Niemelä et al., 2017). Previous studies show a significant positive correlation between STEM education and CT skills (Hsu et al., 2018; Sun et al., 2021a). Using CT and coding to solve a mathematics or science problem is quite different from using CT in a language or art lesson (Australian Computing Academy, 2019).

One non-CS-focused approach is to adopt a cross-domain teaching mode, namely allowing students to work with materials from various domains via computing. This can enable them to deepen their understanding of cross-domain knowledge, experience how cross-domain knowledge and computing can help solve complex real-world problems, and foster interest in studying STEM (Hsu et al., 2018).

While a wide range of new technologies is available for age-appropriate CT development, more research is required to design and develop pedagogies for employing these tools effectively to foster young learners' CT skill development. Furthermore, integrating the development of CT skills with the acquisition of discipline-based content knowledge should help young learners to appreciate the real-world application of CT (Ching et al., 2018; Hsu et al., 2018).

Kale et al. (2018) suggested three strategies that can help teachers make the connections between CT and their teaching of other subjects in K-12 settings: (i) use content-specific examples, (ii) recognise the similarities between CT and the problem you need to solve, (iii) use methods of teaching problem-solving (e.g., modelling).

3.6. Addressing gender and equity when implementing CT in the curriculum

Research on gender balance and equity in the CT field is scarce, and the findings from those studies investigating gender and CT skills are often contradictory (Tikva & Tambouris, 2021). Some studies (Atmatzidou & Demetriadis, 2016; del Olmo-Muñoz et al., 2020; Tsai et al., 2020; Witherspoon et al., 2017; Wu & Su, 2021) find no significant relationship between gender and the acquisition of CT skills, while others conclude that there *are* gender differences in the approach to learning CT (Labusch & Eickelmann, 2020; Román-González et al., 2017; Tomokiyo, 2018; Wei et al., 2021).

The nature of gender differences in CT may depend on the type of problems, tasks or activities proposed for acquiring such skills (Israel-Fishelson et al., 2021; Román-González et al., 2017). Another possible contribution to the diversity of results might also depend on the differences in tools and instruments (i.e., self-report vs practical activities) used to assess the variety of CT-related concepts, which may complicate the current state of the art. For instance, Tsai et al. (2020) found that boys self-report a significantly higher disposition for decomposition thinking than girls. In another study, Rijke et al. (2018) found that "after the age of 9.5 years old, female students begin to outperform their male peers on the abstraction task" (p. 85); when students reach fourth grade, girls are likely to outstrip boys on abstraction. Some authors report a significant interaction effect for gender and age, hypothesising a possible link between this specificity and a gender-related developmental trajectory in consolidating this ability. Given this scenario, the availability of additional and more challenging materials would be desirable, as would adaptation that considers gender differences in the light of different developmental stages. Guggemos (2021) demonstrates that "motivation, in the form of CT self-concept and self-determined motivation, plays an important role in explaining CT level and gender differences," (p. 12) with females showing lower CT self-concept, lower computer literacy, and lower self-determined motivation – findings which translate into a negative association with CT for females.

According to Sun et al. (2021a, p. 355), who conducted an empirical study to explore the association between students' STEM learning attitude and their CT skills through a self-report survey:

> [the] learning attitude of girls from primary school towards STEM was generally more positive than that of boys in the same period, while for CT skills, although the gender difference was not significant, the score of girls was slightly higher than that of boys.

According to Kong et al. (2018), this could be explained by differences in the development of boys and girls at this stage. On the other hand, boys were more interested in programming than girls were, and so "teachers might need to pay attention to the engagement level of girls and employ strategies to enhance their interest in programming" (Kong et al., 2018, p. 188). Some studies (e.g., Balanskat et al., 2018) point out that the integration of CT skills in elective subjects such as Technology, in which usually most of the enrolled students are male, can widen the gap both in terms of gender but also between those interested in programming and those who are not. To address this challenge, the best option appears to make it compulsory for all students to engage in courses where they can learn some basics about CT/programming. In elective subjects, a special effort could be put into recruiting girls and those students who do not think they can engage in programming courses (Balanskat et al., 2018).

Another important matter in this regard, and a possible source for differences in study results, is the variety of approaches and methodologies implied in CT skills support and integration, and the ultimate focus of these, which may also vary depending on cross-cultural differences (Upadhyaya et al., 2020). Leonard et al. (2021) investigated how students choreographing dance performances involving virtual dancers utilise embodied ways of thinking within CT concepts and found that dance provides opportunities for all young people to be engaged in programming. This study showed that, with such integration, students' CT test scores increased significantly and that their embodied thought processes allowed them to enact various computational and choreographic practices. Comparing computational creativity scores regarding personal characteristics has revealed some significant differences between girls and boys. In some cases, mean levels of computational creativity are more remarkable for girls than for boys: "girls were significantly more creative than boys in terms of both creative thinking and computational creativity" (Israel-Fishelson et al., 2021, p. 1436).

Additionally, teachers' role in engaging all students is vital for gender balance and equity. Cateté et al. (2020) state that successful professional development prepares teachers to acquire the skills to teach CS and teach diverse student cohorts with different ethnicities, socioeconomic backgrounds, and genders.

In terms of curriculum enactment, specifically dealing with compulsory CT education for all students, Hsu et al. (2019, p. 268) argue that:

> […] mandatory coursework stems both from the notion that CT is a foundational skill that all students should have to be digitally competent and be active participants in a world where computing is pervasive and from a desire to motivate interest in CS and STEM, especially among girls and underrepresented minorities.

According to Hsu at al. (2019), one example of such initiatives is Code.org in the United States, which aims to increase the participation of women and minorities by implementing CS in the core curriculum. In Ghana and Burkina Faso, the Teach Need Girls mentorship program aims to teach girls how to code and create technology. Additionally, broader participation in computer science, more content responding to the need for diversity in computing, and tackling issues such as equity or accessibility are all highlighted as priorities in the K–12 Computer Science Framework. In the case of schools that already have compulsory CT curricula, implementing additional enrichment programs can enhance interest and motivation, and lead to more in-depth learning. A case study conducted on this topic in the UK found that enrichment programs, for instance Teach Future Girls and Hour of Code, had a positive effect on students, especially on girls, when it came to continuing their studies in higher-level computing courses.

Conversely, the literature review on equity carried out by Huang and Looi (2020) highlights that "unplugged activities appear in curricula that are specifically designed for girls, students of colour, students with special needs, and students in low-income communities, but there have been no studies that theorised the rationale for their inclusion" (p. 97). On the one hand, overall results in this matter point to the variety of tools and approaches as a possible explanation for inconsistencies in results. On the other hand, the results support the need for additional investigations into this matter, accounting for increased consistency in approaches and tools (i.e., use of self-reporting tools in combination with practical tasks about CT). At the same time, results also indicate the need to avoid gender biasing the proposed activities (i.e., proposing girls perform different tasks with respect to boys as a baseline).

4. Conclusions

This paper has discussed significant developments in integration of CT in compulsory education between 2016 and 2021, mainly focusing on CT definitions and curriculum integration approaches as emerged from the analysis of outcomes from the reported literature review. A wider range of evidence regarding CT pedagogies, assessment and professional development of teachers in computing education was also collected through this systematic literature review, which contributed to and is discussed in the study by Bocconi et al. (2022).

Results from the review show we are reaching a plateau in the debate on CT definitions, which is now mainly focusing on the set of key constituent CT concepts, including abstraction, algorithmic thinking, automation, decomposition, debugging, and generalisation. These concepts are correlated with several attitudes and skills (or practices), including creating computational artefacts, testing and debugging, collaboration and creativity, and the capacity to address open-ended problems. From this perspective, CT can be framed as a fundamental competence for a well-informed citizen capable of facing the challenges society continues to pose. CT also offers considerable potential for creative problem-solving and the adoption of innovative approaches in several other subject areas. Therefore, it has a pivotal role to play in compulsory education.

Coding/programming provides a laboratory for teaching and learning CT, for making CT concepts concrete. It can also be a learning tool for investigating other domains or self-expression. However, it is generally agreed that CT entails more than coding or programming (Basu et al., 2020; Barendsen et al., 2016). Although all key CT concepts could in principle be addressed both in primary and lower secondary education, a learning progression is emerging from the actual integration of CT in the curriculum, where the study of programming and algorithms provide the basis for developing CT skills (e.g., Zhang et al., 2020). Starting from sequences of instructions, iterations, conditionals and use of variables in primary school and proceeding to conditionals, operators and data structures in lower secondary. Particular attention is devoted to well-known learner difficulties in primary such as variables initialisation (Franklin et al., 2016), and the challenges of teaching and learning design in programming (Waite et al., 2020). In lower secondary, particular attention is devoted to the theme of abstraction in computing (Grover et al., 2019; Statter & Armoni, 2020).

Concerning curriculum integration, results highlight different implications related to the three main approaches adopted for integrating CT, namely as (i) a cross-curriculum theme, (ii) within other subjects (e.g., mathematics and tech), or (iii) as a separate subject (e.g., CS subject).

The central role played by teachers and the setting of curriculum priorities emerged as key factors from the analysis of the selected papers. The positioning of CT skills in the overall curriculum poses several demands at both policy-making and educational management/organisation levels: making space in the curriculum for including foundational CS concepts to develop CT skills; providing clear guidelines on the amount of time that teachers should devote to teaching basic CS content; allocating adequate resources for developing high-quality instructional material; and sharing examples of sound pedagogical practices. When CT skills are positioned as a cross-curricular theme, it is crucial to clarify the respective responsibilities of each subject teacher in this process.

Open questions for future investigation include (among others) how CT skills are taught and assessed when implementing CT in the curriculum, and how to pursue adequate gender balance and equity.

4.1. Limitations

As with any systematic review, the one presented here has its limitations. First, the search for identifying relevant academic publications was limited to 24 top-tier journals and conference proceedings. Furthermore, although two search strings with several combinations of key terms were applied, if the authors had not included these specific terms in the title, abstract and keywords of their paper/s, the respective article/s may have been excluded from this review.

Finally, certain limitations can be attributed to the screening and inclusion processes, which involved nine researchers, although several measures were taken to ensure inter-rater reliability, as described in Section 2.4 above.

Data availability statement

The following files generated in the context of this study are available on Zenodo:

- The list of the 98 publications analysed in-depth through the review matrix: https://doi.org/10.5281/zenodo.7603313
- The structure of the complete review matrix: https://doi.org/10.5281/zenodo.7603402

References

Arfé, B., Vardanega, T., & Ronconi, L. (2020). The Effects of coding on children's planning and inhibition skills. *Computers and Education, 148,* 1–16. https://doi.org/10.1016/j.compedu.2020.103807

Asbell-Clarke, J., Rowe, E., Almeda, V., Edwards, T., Bardar, E., Gasca, S., Baker, R. S., & Scruggs, R. (2021). The Development of students' computational thinking practices in elementary- and middle-school classes using the learning game, Zoombinis. *Computers in Human Behavior, 115,* 1–14. https://doi.org/10.1016/j.chb.2020.106587

Atmatzidou, S., & Demetriadis, S. (2016). Advancing students' computational thinking skills through educational robotics: A Study on age and gender relevant differences. *Robotics and Autonomous Systems*, *75*, 661–670. https://doi.org/10.1016/j.robot.2015.10.008

Australian Computing Academy. (2019). *Coding and computational thinking—What is the evidence?* Australian Computing Academy. https://education.nsw.gov.au/content/dam/main-education/teaching-and-learning/education-for-a-changing-world/media/documents/Coding-and-Computational-Report_A.pdf

Balanskat, A., Engelhardt, K., & Licht, A. H. (2018). *Strategies to include computational thinking in school curricula in Norway and Sweden—European Schoolnet's 2018 Study Visit*. European Schoolnet. http://www.eun.org/documents/411753/817341/Computational_thinking_report_2018.pdf/4d3d6fa0-dedd-4b62-a201-a26bf4dfd3a0

Barendsen, E., Grgurina, N., & Tolboom, J. (2016). A New informatics curriculum for secondary education in The Netherlands. In A. Brodnik, & F. Tort (Eds.), *Lecture Notes in Computer Science*: *Vol. 9973* (pp. 105–117). Springer. https://doi.org/10.1007/978-3-319-46747-4_9

Basu, S., Rutstein, D., Shear, L., & Xu, Y. (2020). A Principled approach to designing a computational thinking practices assessment for early grades. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education* (pp. 912-918). Association for Computing Machinery. https://doi.org/10.1145/3328778.3366849

Belur, J., Tompson, L., Thornton, A., & Simon, M. (2018). Interrater reliability in systematic review methodology: Exploring variation in coder decision-making. *Sociological Methods & Research*, *50*(2), 837–865. https://doi.org/10.1177/0049124118799372

Bocconi, S., Chioccariello, A., Dettori, G., Ferrari, A., & Engelhardt, K. (2016). *Developing Computational Thinking in Compulsory Education - Implications for policy and practice.* Publications Office of the European Commission. https://doi.org/10.2791/792158

Bocconi, S., Chioccariello, A., Kampylis, P., Dagienė, V., Wastiau, P., Engelhardt, K., Earp, J., Horvath, M.A., Jasutė, E., Malagoli, C., Masiulionytė-Dagienė, V., & Stupurienė, G. (2022). *Reviewing Computational Thinking in Compulsory Education*. Publications Office of the European Union. https://doi.org/10.2760/126955

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 annual meeting of the American educational research association, Vancouver, Canada* (pp. 1–25). https://scratched.gse.harvard.edu/ct/files/AERA2012.pdf

Caeli, E. N., & Yadav, A. (2020). Unplugged approaches to computational thinking: A Historical perspective. *TechTrends, 64*(1), 29–36. https://doi.org/10.1007/s11528-019-00410-5

Cateté, V., Alvarez, L., Isvik, A., Milliken, A., Hill, M., & Barnes, T. (2020). Aligning theory and practice in teacher professional development for computer science. In *Proceedings of the 20th Koli Calling International Conference on Computing Education Research* (pp. 1–11). Association for Computing Machinery. https://doi.org/10.1145/3428029.3428560

Ching, Y.-H., Hsu, Y. C., & Baldwin, S. (2018). Developing Computational thinking with educational technologies for young learners. *TechTrends*, *62*(6), 563–573. https://doi.org/10.1007/s11528-018-0292-7

Coenraad, M., Palmer, J., Weintrop, D., Eatinger, D., Crenshaw, Z., Pham, H., & Franklin, D. (2021). The Effects of providing starter projects in open-ended scratch activities. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education* (pp. 38–44). Association for Computing Machinery. https://doi.org/10.1145/3408877.3432390

Corradini, I., Lodi, M., & Nardelli, E. (2017). Conceptions and misconceptions about computational thinking among Italian primary school teachers. In *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

Corradini, I., Lodi, M., & Nardelli, E. (2018). An Investigation of Italian primary school teachers' view on coding and programming. In S. N. Pozdniakov, & V. Dagiene (Eds.), *Lecture Notes in Computer Science*: *Vol. 11169* (pp. 228–243). Springer. https://doi.org/10.1007/978-3-030-02750-6_18

Curzon, P., Bell, T., Waite, J., & Dorling, M. (2019). Computational thinking. In S. Fincher & A. Robins (Eds.), *The Cambridge Handbook of Computing Education Research* (Cambridge Handbooks in Psychology, pp. 513-546). Cambridge University Press. https://doi.org/10.1017/9781108654555.018

Csizmadia, A., Standl, B., & Waite, J. (2019). Integrating the constructionist learning theory with computational thinking classroom activities. *Informatics in Education*, *18*(1), 41–67. https://doi.org/10.15388/infedu.2019.03

Dagli, Z., & Sancar Tokmak, H. (2021). Exploring high school computer science course teachers' instructional design processes for improving students' "computational thinking" skills. *Journal of Research on Technology in Education*, *54*(4), 511–534. https://doi.org/10.1080/15391523.2021.1881844

del Olmo-Muñoz, J., Cózar-Gutiérrez, R., & González-Calero, J. A. (2020). Computational thinking through unplugged activities in early years of primary education. *Computers and Education, 150,* 1–19. https://doi.org/10.1016/j.compedu.2020.103832

Djambong, T., Freiman, V., Gauvin, S., Paquet, M., & Chiasson, M. (2018). Measurement of computational thinking in K-12 education: The Need for innovative practices. In D. Sampson, D. Ifenthaler, J. Spector, & P. Isaías (Eds.), *Digital technologies: Sustainable innovations for improving teaching and learning* (pp. 193–222). Springer. https://doi.org/10.1007/978-3-319-73417-0_12

Eickelmann, B., Labusch, A., & Vennemann, M. (2019). Computational thinking and problem-solving in the context of IEA-ICILS 2018. In D. Passey, R. Bottino, C. Lewin, & E. Sanchez (Eds.), *IFIP TC 3 Open Conference on Computers in Education, OCCE 2018 Proceedings* (pp. 14–23). Springer. https://doi.org/10.1007/978-3-030-23513-0_2

European Commission. (2020a). *Digital education action plan 2021-2027: Resetting education and training for the digital age.* https://eurlex.europa.eu/legal-content/EN/TXT/?uri=CELEX:52020DC0624

European Commission. (2020b). Commission staff working document accompanying the document communication from the commission to the European Parliament, the Council, the European Economic and Social Committee and the Committee of the regions digital education action plan 2021-2027 resetting education and training for the digital age. https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:52020SC0209&qid=1647943853396

Fagerlund, J., Häkkinen, P., Vesisenaho, M., & Viiri, J. (2020). Assessing 4th grade students' computational thinking through scratch programming projects. *Informatics in Education*, *19*(4), 611–640. https://doi.org/10.15388/infedu.2020.27

Fessakis, G., & Prantsoudi, S. (2019). Computer science teachers' perceptions, beliefs and attitudes on computational thinking in Greece. *Informatics in Education*, *18*(2), 227–258. https://doi.org/10.15388/infedu.2019.11

Forlizzi, L., Lodi, M., Lonati, V., Mirolo, C., Monga, M., Montresor, A., Morpurgo, A., & Nardelli, E. (2018). A Core informatics curriculum for Italian compulsory education. In S. N. Pozdniakov, & V. Dagiene (Eds.), *Lecture Notes in Computer Science*: *Vol. 11169* (pp. 141–153). Springer. https://doi.org/10.1007/978-3-030-02750-6_11

Fraillon, J., Ainley, J., Schulz, W., Duckworth, D., Friedman, T. (2019). Computational thinking framework. In *IEA International Computer and Information Literacy Study 2018 Assessment Framework*. Springer. https://doi.org/10.1007/978-3-030-19389-8_3

Franklin, D., Hill, C., Dwyer, H. A., Hansen, A. K., Iveland, A., & Harlow, D. B. (2016). Initialisation in Scratch: Seeking knowledge transfer. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education* (pp. 217–222). Association for Computing Machinery. https://doi.org/10.1145/2839509.2844569

González-González, C. S., Herrera-González, E., Moreno-Ruiz, L., Reyes-Alonso, N., Hernández-Morales, S., Guzmán-Franco, M. D., & Infante-Moro, A. (2019). Computational thinking and down syndrome: An Exploratory study using the KIBO robot. *Informatics, 6*(2), 1–20. https://doi.org/10.3390/informatics6020025

Grgurina, N., Barendsen, E., Suhre, C., Zwaneveld, B., & Van Veen, K. (2018). Assessment of modeling and simulation in secondary computing science education. In Q. Cutts & A. Muhling (Eds.), *Proceedings of the 13th Workshop in Primary and Secondary Computing Education* (pp. 1-10). Association for Computing Machinery. https://doi.org/10.1145/3265757.3265764

Grover, S., Jackiw, N., & Lundh, P. (2019). Concepts before coding: Non-programming interactives to advance learning of introductory programming concepts in middle school. *Computer Science Education*, *29*(2-3), 106–135. https://doi.org/10.1080/08993408.2019.1568955

Grover, S., & Pea, R. (2018). Computational thinking: A Competency whose time has come. In S. Sentance, E. Barendsen, & S. Carsten (Eds.), *Computer Science Education: Perspectives on Teaching and Learning in School* (pp. 19–38). Bloomsbury. https://doi.org/10.5040/9781350057142.ch-003

Guggemos, J. (2021). On the predictors of computational thinking and its growth at the high-school level. *Computers and Education*, *161*, 1–15. https://doi.org/10.1016/j.compedu.2020.104060

Hallgren, K. A. (2012). Computing inter-rater reliability for observational data: An Overview and tutorial. *Tutorials in Quantitative Methods for Psychology, 8*, 23–34. https://doi.org/10.20982%2Ftqmp.08.1.p023

Hazzan, O., Ragonis, N., & Lapidot, T. (2020). Computational thinking. In O. Hazzan, N. Ragonis, & T. Lapidot (Eds.), *Guide to Teaching Computer Science* (pp. 57–74). Springer. https://doi.org/10.1007/978-3-030-39360-1_4

Hooshyar, D., Pedaste, M., Yang, Y., Malva, L., Hwang, G.-J., Wang, M., Lim, H., & Delev, D. (2020). From gaming to computational thinking: An Adaptive educational computer game-based learning approach. *Journal of Educational Computing Research*, *59*(3), 383–409. https://doi.org/10.1177/0735633120965919

Hromkovič, J., Kohn, T., Komm, D., & Serafini, G. (2016). Combining the power of python with the simplicity of logo for a sustainable computer science education. In A. Brodnik & F. Tort (Eds.), *Lecture Notes in Computer Science: Vol. 9973* (pp. 155–166). Springer. https://doi.org/10.1007/978-3-319-46747-4_13

Hromkovič, J., & Lacher, R. (2017). The Computer science way of thinking in human history and consequences for the design of computer science curricula. In V. Dagiene & A. Hellas (Eds.), *Lecture Notes in Computer Science*: *Vol. 10696* (pp. 3–11). Springer. https://doi.org/10.1007/978-3-319-71483-7_1

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

Hsu, Y.-C., Irie, N. R., & Ching, Y.-H. (2019). Computational Thinking Educational Policy Initiatives (CTEPI) across the globe. *TechTrends, 63*(3), 260–270. https://doi.org/10.1007/s11528-019-00384-4

Huang, W., & Looi, C.-K. (2020). A Critical review of literature on “unplugged” pedagogies in K-12 computer science and computational thinking education. *Computer Science Education*, *31*(1), 83–111. https://doi.org/10.1080/08993408.2020.1789411

Israel, M., Chung, M. Y., Wherfel, Q. M., & Shehab, S. (2020). A Descriptive analysis of academic engagement and collaboration of students with autism during elementary computer science. *Computer Science Education*, *30*(4), 444–468. https://doi.org/10.1080/08993408.2020.1779521

Israel-Fishelson, R., & Hershkovitz, A. (2020). Persistence in a game-based learning environment: the case of elementary school students learning computational thinking. *Journal of Educational Computing Research*, *58*(5), 891–918. https://doi.org/10.1177/0735633119887187

Israel-Fishelson, R., Hershkovitz, A., Eguíluz, A., Garaizar, P., & Guenaga, M. (2021). A Log-based analysis of the associations between creativity and computational thinking. *Journal of Educational Computing Research, 58*(8), 1415–1447. https://doi.org/10.1177/0735633120940954

Jocius, R., Joshi, D., Dong, Y., Robinson, R., Catete, V., Barnes, T., Albert, J., Andrews, A., & Lytl, N. (2020). Code, connect, create: The 3C professional development model to support computational thinking infusion. In *Proceedings of the 51st ACM technical symposium on computer science education* (pp. 971–977). Association for Computing Machinery. https://doi.org/10.1145/3328778.3366797

Kale, U., Akcaoglu, M., Cullen, T., Goh, D., Devine, L., Calvert, N., & Grise, K. (2018). Computational what? Relating computational thinking to teaching. *TechTrends*, *62*(6), 574–584. https://doi.org/10.1007/s11528-018-0290-9

Kert, S. B., Kalelioğlu, F., & Gülbahar, Y. (2019). A Holistic approach for computer science education in secondary schools. *Informatics in Education*, *18*(1), 131–150. https://doi.org/10.15388/infedu.2019.06

Komm, D., Hauser, U., Matter, B., Staub, J., & Trachsler, N. (2020). Computational thinking in small packages. In K. Kori & M. Laanpere (Eds.), *Lecture Notes in Computer Science*: *Vol. 12518* (pp. 170–181). Springer. https://doi.org/10.1007/978-3-030-63212-0_14

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

Kong, S.-C., Lai, M., & Sun, D. (2020). Teacher development in computational thinking: Design and learning outcomes of programming concepts, practices and pedagogy. *Computers and Education*, *151*, 1–19. https://doi.org/10.1016/j.compedu.2020.103872

Labusch, A., & Eickelmann, B. (2020). Computational thinking competences in countries from three different continents in the mirror of students’ characteristics and school learning. In S. C. Kong, H. U. Hoppe, T. C. Hsu, R. H. Huang, B. C. Kuo, K. Y. Li, C. K. Looi, M. Milrad, J. L. Shih, K. F. Sin, K. S. Song, M. Specht, F. Sullivan, & J. Vahrenhold (Eds.), *Proceedings of International Conference on Computational Thinking Education 2020* (pp. 2–7). The Education University of Hong Kong. http://www.eduhk.hk/cte2020/doc/CTE2020%20Proceedings.pdf

Leonard, A. E., Daily, S. B., Jörg, S., & Babu, S. V. (2021). Coding moves: Design and research of teaching computational thinking through dance choreography and virtual interactions. *Journal of Research on Technology in Education*, *53*(2), 159–177. https://doi.org/10.1080/15391523.2020.1760754

Li, Q. (2020). Computational thinking and teacher education: An Expert interview study. *Human Behavior and Emerging Technologies*, *3*, 324–338. https://doi.org/10.1002/hbe2.224

Liebe, C., & Camp, T. (2019). An Examination of abstraction in K-12 computer science education. In *Proceedings of the 19th Koli Calling International Conference on Computing Education Research* (pp. 1–9). Association for Computing Machinery. https://doi.org/10.1145/3364510.3364526

Metcalf, S. J., Reilly, J. M., Jeon, S., Wang, A., Pyers, A., Brennan, K., & Dede, C. (2021). Assessing computational thinking through the lenses of functionality and computational fluency. *Computer Science Education*, *31*(2), 199–223. https://doi.org/10.1080/08993408.2020.1866932

Niemelä, P., Partanen, T., Harsu, M., Leppänen, L., & Ihantola, P. (2017). Computational thinking as an emergent learning trajectory of mathematics. In *Proceedings of the 17th Koli Calling international conference on computing education research* (pp. 70-79). Association for Computing Machinery. https://doi.org/10.1145/3141880.3141885

Page, M. J., McKenzie, J. E., Bossuyt, P. M., Boutron, I., Hoffmann, T. C., Mulrow, C. D., Shamseer, L., Tetzlaff, J. M., Akl, E. A., Brennan, S. E., Chou, R., Glanville, J., Grimshaw, J. M., Hrobjartsson, A., Lalu, M. M., Li, T., Loder, E. W., Mayo-Wilson, E., McDonald, S., … Moher, D. (2021). The PRISMA 2020 statement: An Updated guideline for reporting systematic reviews. *The BMJ*, *372*(71), 1–8. https://doi.org/10.1136/bmj.n71

Palts, T., & Pedaste, M. (2020). A Model for developing computational thinking skills. *Informatics in Education*, *19*(1), 113–128. https://doi.org/10.15388/infedu.2020.06

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

Pasternak, A. (2016). Contextualized teaching in the lower secondary education long-term evaluation of a cs course from grade 6 to 10. In *Proceedings of the 47th ACM technical symposium on computing science education* (pp. 657–662). Association for Computing Machinery. https://doi.org/10.1145/2839509.2844592

Pérez-Marín, D., Hijón-Neira, R., Bacelo, A., & Pizarro, C. (2020). Can computational thinking be improved by using a methodology based on metaphors and scratch to teach computer programming to children? *Computers in Human Behavior*, *105*, 1–10. https://doi.org/10.1016/j.chb.2018.12.027

Razak, S., Khan, S., Hussein, N., Alshikhabobakr, H., Gedawy, H., & Yousaf, A. W. (2021). integrating computer science and ICT concepts in a cohesive curriculum for middle school-an experience report. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education* (pp. 966–972). Association for Computing Machinery. https://doi.org/10.1145/3408877.3432528

Repenning, A., Lamprou, A., & Basawapatna, A. (2021). Computing effect sizes of a science-first-then-didactics computational thinking module for preservice elementary school teachers. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education* (pp. 274–280). Association for Computing Machinery. https://doi.org/10.1145/3408877.3432446

Resnick, M. (2017). *Lifelong kindergarten: Cultivating creativity through projects, passion, peers, and play*. The MIT Press.

Rich, P. J., Mason, S. L., & O'Leary, J. (2021). Measuring the effect of continuous professional development on elementary teachers' self-efficacy to teach coding and computational thinking. *Computers and Education*, *168,* 1–15. https://doi.org/10.1016/j.compedu.2021.104196

Rijke, W. J., Bollen, L., Eysink, T. H. S., & Tolboom, J. L. J. (2018). Computational thinking in primary school: An Examination of abstraction and decomposition in different age groups. *Informatics in Education*, *17*(1), 77–92. https://doi.org/10.15388/infedu.2018.05

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

Sherwood, H., Yan, W., Liu, R., Martin, W., Adair, A., Fancsali, C., Rivera-Cash, E., Pierce, M., & Israel, M. (2021). Diverse approaches to school-wide computational thinking integration at the elementary grades: A Cross-case analysis. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education* (pp. 253–259). Association for Computing Machinery. https://doi.org/10.1145/3408877.3432379

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

Statter, D., & Armoni, M. (2020). Teaching abstraction in computer science to 7th grade students. *ACM Transactions on Computing Education*, *20*(1), 1–37. https://doi.org/10.1145/3372143

Sun, L., Hu, L., Yang, W., Zhou, D., & Wang, X. (2021a). STEM learning attitude predicts computational thinking skills among primary school students. *Journal of Computer Assisted Learning*, *37*(2), 346–358. https://doi.org/10.1111/jcal.12493

Sun, L., Hu, L., & Zhou, D. (2021b). Which way of design programming activities is more effective to promote K- 12 students' computational thinking skills? A Meta-analysis. *Journal of Computer Assisted Learning*, *37*(4), 1048–1062. https://doi.org/10.1111/jcal.12545

Tang, X., Yin, Y., Lin, Q., Hadad, R., & Zhai, X. (2020). Assessing computational thinking: A Systematic review of empirical studies. *Computers and Education*, *148,* 1–22. https://doi.org/10.1016/j.compedu.2019.103798

Taslibeyaz, E., Kursun, E., & Karaman, S. (2020). How to develop computational thinking: A Systematic review of empirical studies. *Informatics in Education*, *19*(4), 701–719. https://doi.org/10.15388/INFEDU.2020.30

Tedre, M., & Denning, P. J. (2016). The Long quest for computational thinking. In *Proceedings of the 16th Koli Calling international conference on computing education research* (pp. 120–129). Association for Computing Machinery. https://doi.org/10.1145/2999541.2999542

Tikva, C., & Tambouris, E. (2021). Mapping computational thinking through programming in K-12 education: A Conceptual model based on a systematic literature review. *Computers and Education, 162*, 1–23. https://doi.org/10.1016/j.compedu.2020.104083

Tomokiyo, L. M. (2018). Successes and challenges in implementing a progressive K-8 computer science curriculum. In Q. Cutts, & A. Muhling (Eds.), *Proceedings of the 13th Workshop in Primary and Secondary Computing Education* (pp. 1–6). Association for Computing Machinery. https://doi.org/10.1145/3265757.3265779

Tonbuloğlu, B., & Tonbuloğlu, I. (2019). The Effect of unplugged coding activities on computational thinking skills of middle school students. *Informatics in Education*, *18*(2), 403–426. https://doi.org/10.15388/infedu.2019.19

Tsai, M.-J., Liang, J.-C., & Hsu, C.-Y. (2020). The Computational thinking scale for computer literacy education. *Journal of Educational Computing Research*, *59*(4), 579–602. https://doi.org/10.1177/0735633120972356

UNESCO, & IFIP TC3. (2019). *Coding, programming and the changing curriculum for computing in schools*. https://wcce2022.org/pubs/UNESCO%20meeting%20at%20OCCE%202018%20report%20final.pdf

Upadhyaya, B., McGill, M. M., & Decker, A. (2020). A Longitudinal analysis of K-12 computing education research in the United States: Implications and recommendations for change. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education* (pp. 605–611). Association for Computing Machinery. https://doi.org/10.1145/3328778.3366809

Waite, J., Curzon, P., Marsh, W., & Sentance, S. (2020). Difficulties with design: The Challenges of teaching design in K-5 programming. *Computers and Education*, *150,* 1–28. https://doi.org/10.1016/j.compedu.2020.103838

Webb, M., Davis, N., Bell, T., Katz, Y., Reynolds, N., Chambers, D. P., & Sysło, M. M. (2017). Computer science in K-12 school curricula of the 2lst century: Why, what, and when? *Education and Information Technologies*, *22*(2), 445–468. https://doi.org/10.1007/s10639-016-9493-x

Wei, X., Lin, L., Meng, N., Tan, W., Kong, S.-C., & Kinshuk. (2021). The Effectiveness of partial pair programming on elementary school students' Computational Thinking skills and self-efficacy. *Computers and Education*, *160,* 1–15. https://doi.org/10.1016/j.compedu.2020.104023

Weintrop, D., & Wilensky, U. (2019). Transitioning from introductory block-based and text-based environments to professional programming languages in high school computer science classrooms. *Computers and Education*, *142*, 1–17. https://doi.org/10.1016/j.compedu.2019.103646

Wing, J. (2017). Computational thinking's influence on research and education for all. *Italian Journal of Educational Technology*, *25*(2), 7–14. https://doi.org/10.17471/2499-4324/922

Witherspoon, E. B., Higashi, R. M., Schunn, C. D., Baehr, E. C., & Shoop, R. (2017). Developing computational thinking through a virtual robotics programming curriculum. *ACM Transactions on Computing Education*, *18*(1), 1-20. https://doi.org/10.1145/3104982

Wu, S.-Y., & Su, Y.-S. (2021). Visual programming environments and computational thinking performance of fifth- and sixth-grade students. *Journal of Educational Computing Research*, *59*(6), 1075–1092. https://doi.org/10.1177/0735633120988807

Yağcı, M. (2019). A Valid and reliable tool for examining computational thinking skills. *Education and Information Technologies*, *24*(1), 929–951. https://doi.org/10.1007/s10639-018-9801-8

Zhang, L., & Nouri, J. (2019). A Systematic review of learning computational thinking through Scratch in K-9. *Computers and Education*, *141,* 1–25. https://doi.org/10.1016/j.compedu.2019.103607

Zhang, L. C., Nouri, J., & Rolandsson, L. (2020). Progression of computational thinking skills in Swedish compulsory schools with block-based programming. In *Proceedings of the Twenty-Second Australasian Computing Education Conference* (pp. 66–75). Association for Computing Machinery. https://doi.org/10.1145/3373165.3373173

Appendix 1. An excerpt from the review matrix focusing on the three research questions

The structure of the complete review matrix is available in Zenodo: https://doi.org/10.5281/zenodo.7603402. The complete references are also available in Zenodo: https://doi.org/10.5281/zenodo.7603313.

RQ1 - CT definition and terminology
RQ2 - Implementation (How things are done or proposed approach)
RQ3- How can gender balance and equity be ensured when implementing CT in the curriculum?

No.	Short reference	RQ1	RQ2	RQ3
	Academic literature			
1	Arfé, Vardanega & Ronconi (2020)	🟦	🟩	
2	Asbell-Clarke, Rowe, Almeda, Edwards, Bardar, Gasca, Baker & Scruggs (2021)	🟦	🟩	
3	Barendsen, Grgurina & Tolboom (2016)	🟦	🟩	
4	Basu, Rutstein, Shear & Xu (2020)		🟩	
5	Caeli & Yadav (2020)	🟦	🟩	
6	Cateté, Alvarez, Isvik, Milliken, Hill & Barnes (2020)			🟨
7	Ching, Hsu & Baldwin (2018)		🟩	
8	Coenraad, Palmer, Weintrop, Eatinger, Crenshaw, Pham & Franklin (2021)		🟩	
9	Corradini, Lodi, & Nardelli (2017)	🟦		
10	Corradini, Lodi & Nardelli (2018)	🟦	🟩	
11	Csizmadia, Standl, & Waite (2019)	🟦	🟩	
12	Curzon, Bell, Waite & Dorling (2019)	🟦	🟩	
13	Dagli & Sancar Tokmak (2021)		🟩	
14	del Olmo-Muñoz, Cózar-Gutiérrez & González-Calero (2020)		🟩	🟨
15	Djambong, Freiman, Gauvin, Paquet & Chiasson (2018)		🟩	
16	Eickelmann, Labusch & Vennemann (2019)	🟦	🟩	
17	Fagerlund, Häkkinen, Vesisenaho & Viiri (2020)		🟩	
18	Fessakis & Prantsoudi (2019)	🟦	🟩	
19	Forlizzi, Lodi, Lonati, Mirolo, Monga, Montresor, Morpurgo & Nardelli (2018)		🟩	
20	Franklin, Hill, Dwyer, Hansen, Iveland, & Harlow (2016)		🟩	
21	González-González, Herrera-González, Moreno-Ruiz, Reyes-Alonso, Hernández-Morales, Guzmán-Franco & Infante-Moro (2019)		🟩	
22	Grgurina, Barendsen, Suhre, Zwaneveld & Van Veen (2018)	🟦	🟩	
23	Grover & Pea (2018)	🟦	🟩	
24	Grover, Jackiw & Lundh (2019)		🟩	
25	Guggemos (2021)			🟨
26	Hazzan, Ragonis & Lapidot (2020)	🟦		
27	Hooshyar, Pedaste, Yang, Malva, Hwang, Wang, Lim & Delev (2020)	🟦	🟩	
28	Hromkovič, Kohn, Komm & Serafini (2016)		🟩	
29	Hromkovič & Lacher (2017)		🟩	
30	Hsu, Chang & Hung (2018)		🟩	
31	Hsu, Irie & Ching (2019)		🟩	🟨
32	Huang & Looi (2020)	🟦	🟩	🟨
33	Israel, Chung, Wherfel & Shehab (2020)		🟩	
34	Israel-Fishelson & Hershkovitz (2020)	🟦	🟩	
35	Israel-Fishelson, Hershkovitz, Eguíluz, Garaizar & Guenaga (2021)		🟩	🟨
36	Jocius, Joshi, Dong, Robinson, Catete, Barnes, Albert, Andrews & Lytl (2020)	🟦	🟩	
37	Kale, Akcaoglu, Cullen, Goh, Devine, Calvert & Grise (2018)	🟦	🟩	
38	Kert, Kalelioğlu & Gülbahar (2019)		🟩	
39	Komm, Hauser, Matter, Staub & Trachsler (2020)	🟦		
40	Kong, Chiu & Lai (2018)			🟨
41	Kong, Lai & Sun (2020)	🟦	🟩	
42	Labusch & Eickelmann (2020)			🟨
43	Leonard, Daily, Jörg & Babu (2021)	🟦		🟨
44	Li (2020)		🟩	
45	Liebe & Camp (2019)		🟩	
46	Metcalf, Reilly, Jeon, Wang, Pyers, Brennan & Dede (2021)		🟩	
47	Niemelä, Partanen, Harsu, Leppänen & Ihantola (2017)		🟩	
48	Palts & Pedaste (2020)		🟩	
49	Pasternak (2016)		🟩	

50	Pérez-Marín, Hijón-Neira, Bacelo, & Pizarro (2020)	🟦	🟩	
51	Razak, Khan, Hussein, Alshikhabobakr, Gedawy & Yousaf (2021)		🟩	
52	Repenning, Lamprou & Basawapatna (2021)		🟩	
53	Rich, Mason & O'Leary (2021)	🟦	🟩	
54	Rijke, Bollen, Eysink & Tolboom (2018)		🟩	🟨
55	Román-González, Moreno-León & Robles (2019)		🟩	
56	Román-González, Pérez-González & Jiménez-Fernández (2017)	🟦	🟩	🟨
57	Sáez-López, Román-González & Vázquez-Cano (2016)		🟩	
58	Sherwood, Yan, Liu, Martin, Adair, Fancsali, Rivera-Cash, Pierce & Israel (2021)		🟩	
59	Statter & Armoni (2020)		🟩	
60	Sun, Hu, Yang, Zhou & Wang (2021)		🟩	🟨
61	Sun, Hu & Zhou (2021)		🟩	
62	Tang, Yin, Lin, Hadad & Zhai (2020)	🟦		
63	Taslibeyaz, Kursun & Karaman (2020)	🟦		
64	Tedre & Denning (2016)	🟦		
65	Tikva & Tambouris (2021)	🟦	🟩	🟨
66	Tomokiyo (2018)		🟩	🟨
67	Tonbuloğlu & Tonbuloğlu (2019)		🟩	
68	Tsai, Liang & Hsu (2020)	🟦		🟨
69	Upadhyaya, McGill & Decker (2020)	🟦		🟨
70	Waite, Curzon, Marsh & Sentance (2020)		🟩	
71	Webb, Davis, Bell, Katz, Reynolds, Chambers, & Sysło (2017)	🟦	🟩	
72	Wei, Lin, Meng, Tan, Kong & Kinshuk (2021)		🟩	🟨
73	Weintrop & Wilensky (2019)	🟦		
74	Witherspoon, Higashi, Schunn, Baehr & Shoop (2017)			🟨
75	Wu & Su (2021)		🟩	🟨
76	Yağcı (2019)		🟩	
77	Zhang & Nouri (2019)	🟦	🟩	
78	Zhang, Nouri & Rolandsson (2020)		🟩	
Grey literature				
1	Australian Computing Academy (2019)	🟦	🟩	
2	Balanskat, Engelhardt & Licht (2018)		🟩	🟨
3	Bocconi, Chioccariello & Earp (2018)	🟦	🟩	
4	Dagienė, Graželienė, Jasiukevičienė, Jasutė, Skūpas, Tamošiūnaitė, Žandaris, Balvočius & Leonavičius (2021)	🟦		
5	Crick (2017)	🟦		
6	Fiorin, Bettini, Cerini, Cicatelli, Da Re, Langé, Lorenzoni, Nigris, Petracca, Rossi, Silvestro, Ventre, Zan, Marrocchi (2017)	🟦		
7	Fraillon, Ainley, Schulz, Duckworth & Friedman (2019)	🟦		
8	Government of Ireland (2018)	🟦	🟩	
9	Himpsl-Gutermann, Brandhofer, Frick, Fikisz, Steiner, Bachinger, Gawin, Gawin, Szepannek & Lechner (2018)	🟦	🟩	
10	Kunnskapsdepartementet (2017)		🟩	
11	Kunnskapsdepartementet (2020)		🟩	
12	Ministère de l'éducation nationale et de la jeunesse et des sports (2019)		🟩	
13	Ministerio de Educación y Formación Profesional de España (2018)		🟩	
14	Ministerio de Educación y Formación Profesional de España (2019)		🟩	🟨
15	Ministry of Education - Poland (2020)	🟦		
16	Ministry of Education and Research - Romania (2021)		🟩	
17	Ministry of Education -Singapore (2020)	🟦	🟩	
18	Ministry of Science and Education – Croatia (2018)	🟦	🟩	
19	Paniagua & Istance (2018)	🟦		
20	Storte, Webb, Bottino, Passey, Kalas, Bescherer, Smith, Angeli, Katz, Micheuz, Røsvik, Brinda, Fluck, Magenheim, Anderson & Fuschek (2019)		🟩	