OECD Education Working Papers No. 274

The state of the field of computational thinking in early childhood education

Marina Umaschi Bers,
Amanda Strawhacker,
Amanda Sullivan

https://dx.doi.org/10.1787/3354387a-en

OECD

Organisation for Economic Co-operation and Development

EDU/WKP(2022)12

For Official Use **English - Or. English**

5 July 2022

DIRECTORATE FOR EDUCATION AND SKILLS

The State of the Field of Computational Thinking in Early Childhood Education

OECD Education Working Paper No. 274

By Marina Umaschi Bers, Amanda Strawhacker, and Amanda Sullivan (DevTEch Research Group, Tufts University).

This working paper has been authorised by Andreas Schleicher, Director of the Directorate for Education and Skills, OECD.

Marina Bers, Tufts University, marina.bers@tufts.edu

JT03498895

OECD EDUCATION WORKING PAPERS SERIES

Comment on the series is welcome, and should be sent to edu.contact@oecd.org.

This working paper has been authorised by Andreas Schleicher, Director of the Directorate for Education and Skills, OECD.

---

www.oecd.org/edu/workingpapers

Acknowledgements

The authors thank members of the OECD Network on Early Childhood Education and Care (ECEC) and the OECD team for their feedback on initial drafts of this review. A special thanks to Olivia Tighe and Mernie Graziotin for editorial support.

Abstract

Computer programming and associated Computational Thinking (CT) skills are essential to thriving in today's academic and professional world. There has been a growing focus globally on fostering CT skills as well as on introducing computer programming concepts and languages beginning as early as kindergarten and pre-primary school. Tools, curriculum, and frameworks to promote CT in the early years must be designed and implemented in ways that engage children who cannot yet read and write, who learn through play, and who have a short attention span and limited working memory but also strong natural curiosity. This review summarises empirical and theoretical literature on the state of the field of CT as it relates to early learning and development, a time when young children are being introduced to foundational skills, such as literacy and numeracy, which can carefully be complemented by an exploration of CT.

Table of contents

FIGURES

TABLES

1. Introduction

Computer programming powers the global digital environment in which children are growing up today. Websites, smartphone applications, computer games, and even modern microwaves, cars, and vacuum cleaners, all run on code. But how do we write code? And how can young children growing up in today's digital landscape become literate in coding and computer science? Relevant answers to these questions relate to a process called Computational Thinking (hereafter, CT). Along with being crucial to coding and computer science in general, CT is an important skill set across many academic and professional domains (Wing, 2006[1]) (Wing, 2011[2]) (Bers, 2021[3]). CT, which fosters analytical problem solving along with creative expression, is the driving force behind new initiatives focused on introducing young children to programming (Bers, 2021[3]). This review summarises the state of the field of CT as it relates to early learning and development, highlighting empirical research, theoretical and pedagogical work, curricular initiatives, as well as commercially available products and media for supporting CT in young children (ages 3-8).

This document begins with providing key definitions of terms related to CT and background on the field of CT. It goes on to discuss how CT found its place in learning standards and frameworks for early levels of education, as well as research on CT in early learning and development. Next, the review highlights various tools, technologies, and media that have been developed in the past decade for supporting CT in young children, including unplugged and screen-free interfaces. Finally, the review discusses the implementation of CT programmes in OECD countries and breaks down important issues of equity and access in CT education.

Computer programming is becoming an essential skill in the 21st century. Each month, there are an estimated 500,000 openings for computing jobs in the US alone, and a lack of adequately trained people to fill them (Code.org, 2018[4]; Fayer, Lacey and Watson, 2017[5]). A recent forecast from the World Economic Forum listed computer science-related jobs such as data and AI (artificial intelligence), machine learning, software developers, and robotics engineers as the world's fastest growing industries in 2025 (World Economic Forum, 2020[6]). CT skills such as analytical thinking, complex problem solving, and technology innovation and design are among the fastest growing gaps in skilled employees. However, the rationale for supporting the introduction of computer science and CT starting in kindergarten is not limited to the creation of the future workforce, but concerns also the promotion of the future citizenry (Bers, 2021[3]).

The goal of this review is to provide an overview of recent and evidence-based recommendations, trends, and initiatives to inform effective policy decisions for OECD countries to maximise investments in CT education for their citizens, starting with their youngest members. While the development of CT skills in early childhood is a burgeoning field of research, more robust evidence on the diverse tools and approaches that have emerged in recent years is still required to inform policy and practice, in particular to assess the potential benefits and downsides of different CT educational programmes before they may be introduced at scale.

2. Defining Computational thinking, Computer science, and programming

2.1. Definitions

This section defines three key terms necessary for engaging with the literature around CT in the field of learning and education: computer science, computer programming, and computational thinking.

2.2. Computer science

According to the Association for Computer Machinery (ACM), computer science is the study of computers and their algorithmic processes (Tucker, 2003[7]). Computer scientists design experimental algorithms, theorise about why they work, and use those theories to inform new designs and data structures (Dodig-Crnkovic, 2002[8]). The field of computer science encompasses a range of careers and academic concentrations, including artificial intelligence, computer systems and networks, security, database systems, human computer interaction, vision and graphics, numerical analysis, programming languages, software engineering, and theory of computing. While programming is just one element of the vast field of computer science, several core programming concepts are particularly relevant to the development of CT in early learning. Perhaps the most relevant are algorithms, or sequences of commands in which the order matters, and control structures, or instructional commands that deal with the behaviour of algorithms (e.g. a "repeat" loop and a conditional "if-else" statement are both control structures) (Bers, 2018[9]).

2.3. Computer programming

A programme is a series of instructions for a computer or machine to carry out (Code.org, 2021[10]). If computational logic is used to plan programmes, then coding is the process of writing that plan in a programming language that a computer can understand. Programming is used as a tool to create products that reflect a wide range of interests and needs. Any machine that interacts with its environment and functions without an engineer or user controlling its actions, from automatic doors to the Mars Rover, has been coded to behave that way by a computer programmer.

A programmer writes code, a sequence of instructions in a programme. For instance, if dialogue is not sequenced correctly when programming a simple animated story between two characters, the story will not make sense. If the commands to programme a robot are not in the correct order, the robot will not complete the task desired. Control structures specify the order in which sequenced instructions are executed within a programme. Repeat loops, a type of control structure, allow for the repetition of a code sequence multiple times. For example, in a musical programme to play a favourite song, a repeat loop may be used to repeat the chorus of the song multiple times.

2.4. Computational thinking

Although computational thinking (CT) has received considerable attention over the past several years, there is little agreement on what a definition for CT might encompass (Barr and Stephenson, 2011[11]; Grover and Pea, 2013[12]; Guzdial, 2008[13]; National Research Council, 2010[14]; Relkin, 2018[15]; Relkin and Bers, 2019[16]; Shute, Sun and Asbell-Clarke, 2017[17]). The notion of CT encompasses a broad set of analytic and problem solving skills, dispositions, habits, and approaches most often used in computer science, but that can serve

in multiple other contexts (Barr, Harrison and Conery, 2011[18]; Barr and Stephenson, 2011[11]; Lee et al., 2011[19]).

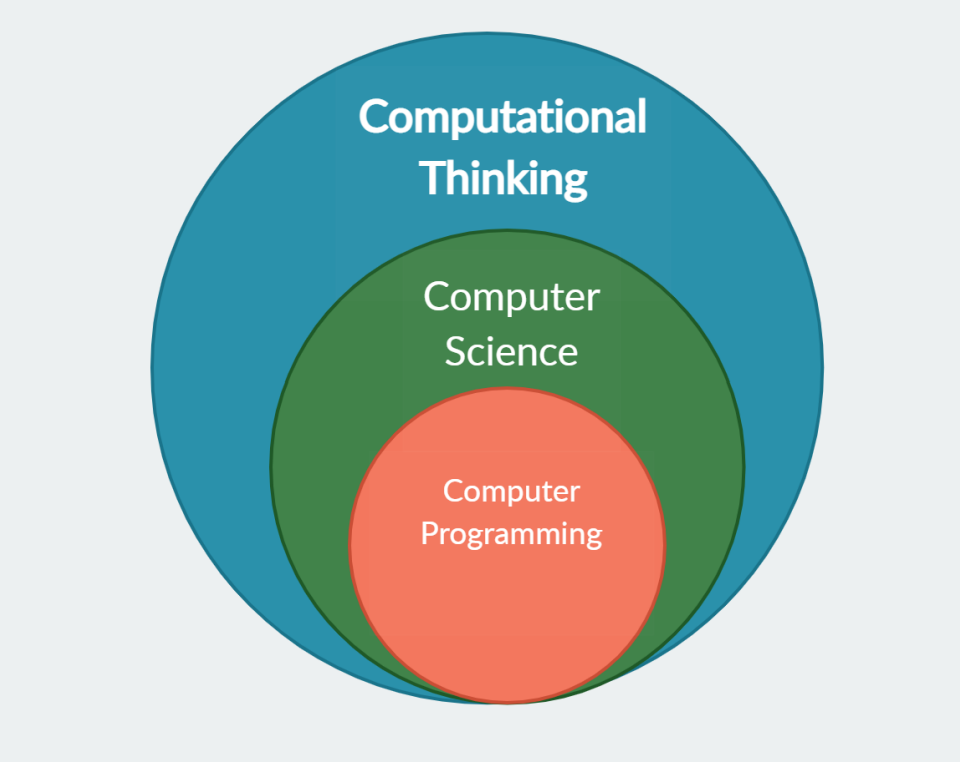
One commonly used definition is that CT describes the thought processes involved in formulating problems and in constructing and/or decomposing the sequential steps of a solution in a form that can be executed by a computer, a human, or a combination of both (Aho, 2011[20]; Kim and Lee, 2016[21]; Wing, 2011[2]). In this way, CT represents a type of analytical thinking that shares similarities with mathematics thinking (e.g. problem solving), engineering thinking (designing and evaluating processes), and scientific thinking (systematic analysis) (Bers, 2010[22]; Bers, 2021[3]).

Mastery of CT includes the processes of pattern recognition, conceptualisation, planning and problem solving. Researchers have found evidence that learning to code can improve children's acquisition of CT skills, perhaps because the act of coding requires logical reasoning that itself relies on sequence and structure (Fraillon et al., 2020[23]; Grover and Pea, 2013[12]; Lye and Koh, 2014[24]; Relkin and Bers, 2020[25]). For this reason, coding and computer programming tools and curriculum and activities that involve logical thinking and sequencing are the most common and accessible ways that educators can begin fostering CT in children.

2.4.1. Summary

Figure 1 illustrates the relationship between the key terms that were defined in this section. Computational Thinking (CT) encompasses a broad set of skills involved in constructing and/or decomposing the sequential steps of a task so that it can be carried out by a computer. CT skills include pattern recognition, conceptualisation, planning, and problem solving, among others, and are used in creative and expressive tasks across computer science disciplines, as well as non-technical disciplines such as mathematics and writing. Computer programming is just one aspect of computer science. A computer programme is a series of instructions for a computer or machine to carry out. These programmes are written by humans (programmers).

Figure 1.Relationship between computational thinking, computer science, and programming



Throughout this review, these terms are explored in the context of studies primarily concerning children aged 3-to-8, because this age span represents a critical time in development when it comes to fostering CT and computer science education. The review focuses on evidence regarding this age range that was collected in early childhood education and care (ECEC) settings. This includes all arrangements providing care and education for children under compulsory school age, regardless of setting, funding, opening hours or programme content (depending on the international context, research may also refer to formal school settings specifically for children in the older end of the intended age range). Some research shows that the economic and developmental impact of interventions that begin in early childhood tend to be associated with lower costs and more durable effects than interventions that begin later (Cunha and Heckman, 2007[26]) (Heckman and Masterov, 2007[27]) (National Research Council, 2001[28]) (Shonkoff and Phillips, 2000[29]). From a teaching and learning perspective, CT concepts, particularly abstract ones, can face a steep learning curve in older students that may be avoided by introducing foundational CT earlier on. Some research has even suggested that age 10-11 could represent a developmental critical period for foundational CT skills (aligned with other critical periods in cognitive development), suggesting the importance of early experiences before this developmental window closes (Lerner and Steinberg, 2009[30]; Sun et al., 2020[31]). Thus, if promoting CT is important in our rapidly developing information age, there are strong arguments for introducing it during the early years of children's education. Furthermore, it is critical that pedagogical approaches and technologies used to introduce CT are consistent with developmentally appropriate practice (Bredekamp, 1992[32]), and that they embrace the maturational stages of children by inviting play, discovery, socialisation, and creativity (Bers, 2018[9]). This review highlights developmentally appropriate practice as it relates to digital technologies, CT, and young children.

2.5. Computational thinking concepts for young children

Computational thinking includes mental processes such as thinking recursively, using abstraction when figuring out a complex task, and applying heuristic reasoning to discover a solution and to identify potential "bugs", or problems. Wing (2006[1]) asserts that just as the printing press facilitated the spread of the three 'R's (reading, writing, and arithmetic), computers facilitate the spread of computational thinking. For this reason, it is vital that all children, regardless of their background and gender, have an equal opportunity to acquire CT skills.

To address pressing equity issues surrounding CT gaps, research suggests that beginning CT education in early childhood may avoid future challenges associated with introducing these skills in later years (Relkin and Bers, 2021[33]). One study of 169 students in Greece aged 15 and 18 found that secondary school students can eventually master CT concepts while engaged in an NXT LEGO Robotics curriculum focused on hands-on experiences followed by oral or written demonstrations of how mechanical systems function. However, the study also highlights the importance of allowing adequate time to attain CT skills, particularly more abstract ones, with female students in their sample requiring more time to achieve equal mastery of several CT skills (Atmatzidou and Demetriadis, 2016[34]). Developmentally, secondary level students have already internalised several assumptions and gender-based stereotypes about academic subjects that can inhibit performance, and research in early childhood suggests that this stereotype threat can be mitigated by early exposure to CT experiences (Sullivan, 2019[35]; Atmatzidou and Demetriadis, 2016[34]) also found that in general, students showed dramatic gains in understanding near the end of the learning unit, suggesting that truncated experiences (such as Hour of Code) may not achieve the same level of educational enrichment.

In later childhood and elementary years, researchers looking at illustrative examples from three National Science Foundation-funded informal education programmes serving 10-18 year-olds in the United States found that scaffolded experiences supported by intensive staff support successfully helped learners to progress from simple tool-use practices, such as debugging and testing, through deeper modification and eventually creation of mechanical systems, resulting in analysis of models designed for real-world applications (Lee et al., 2011[19]; Shonkoff and Phillips, 2000[29]). However, researchers identified barriers in the feasibility of translating CT experiences into classrooms settings, including limits on instructional time and challenges in teacher preparation. In contrast, the emphasis of ECEC programmes on creative and exploratory time with hands-on object manipulation lends itself to early accessibility to technological tools, and to developmental readiness for beginning CT skills. Thus, early intervention might save precious instructional time by preparing students with foundational CT awareness to begin sooner with technology-supported design and creation in older years. This leads to the questions of what does exploring CT look like during the early childhood years, and what are the CT skills that young children can master at an early age. Bers (2020[36]) describes seven "powerful ideas"[1] from CT that are developmentally appropriate for young children to master and that are not tied to a particular computer programming environment, but instead to the discipline of computer science and its associated habits of mind. These ideas are: algorithms, modularity, control structures, representation, hardware/software, the design

[1] The term "powerful ideas" was first coined by Seymour Papert, who described them as new ways of thinking, new ways of putting knowledge to use, and new ways of making personal and epistemological connections with other domains of knowledge (Papert, 2000[273]).

process, and debugging. Table 1 defines these concepts and illustrates how each of them relates to foundational early childhood skills and development.

Table 1. Aligning computational thinking with early childhood foundational skills

Computational Thinking Concepts	Foundational Skills
Algorithms – A series of ordered steps taken in sequence.	Order matters Logical organisation
Modularity – Breaking down complex tasks and procedures into simpler, manageable units.	Breaking down a large task
Control Structures – Controlling the sequence in which a programme is executed. Making decisions based on conditions.	Pattern Recognition
Representation – Concepts can be represented by symbols.	Symbolic representation of letters and numbers
Hardware/Software systems- Computing systems need both hardware and software to operate.	Recognising objects and processes that are human engineered
Design Process – An iterative process used to develop programmes and artefacts with multiple steps.	Writing Process Scientific Method Creative Process
Debugging – Fixing problems in our programmes in a systematic way.	Perseverance Problem solving

Source: Bers (2020[36]), *Coding as a playground: Programming and computational thinking in the early childhood classroom*, Routledge, New York, https://doi.org/10.4324/9781003022602.

3. Computational thinking frameworks and learning standards

3.1. History of CT in learning standards and frameworks

The United Kingdom was one of the first OECD countries to make an international mark with a focus on computing education in the National Curriculum. In 1981, Computer Studies became common for students aged 11–16 in the United Kingdom. The importance of computer studies was recognised, and it later became a compulsory part of the National Curriculum and in 2013, at which time it became a requirement for all students over the age of 4. The UK's approach to computer science education is particularly noteworthy, as the UK's National Computing programme (Department for Education, 2013[37])became the

inspiration for the technology and computing curricula later implemented in the United States, Australia and New Zealand (New South Wales Department of Education, 2019[38]).

Within the United States, the computing curriculum has traditionally been linked with STEM (Science, Technology, Engineering, and Mathematics) disciplines. The STEM acronym came into the American consciousness in the 1950s as a response to the need for a technically oriented workforce, and to maintain national security. In 1958, during the height of the space race, the United States passed the National Defense Education Act (NDEA), which provided funding and incentives for schools to improve their math, science, and engineering curricula to prepare the future workforce. The act also had provisions for research and experimentation in the use of television, radio, and motion pictures for educational purposes. As the cold war ended, the emphasis on national security diminished and the perceived urgency to teach a foreign language dropped, but the need for economic competitiveness remained. With a rapidly growing technological society, learning computer programming provided increased career opportunities. However, computer programming was mainly seen as part of the skillset for mathematicians, scientists, and engineers. Thus, the teaching of computer science drew from methodologies already used in STEM disciplines such as solving pre-set challenges and engaging in competitions. At that time, the broader benefits for everyone to learn how to code could not yet be perceived, as computers did not play a major role in everyday life. In fact, it was not until the past decade that coding and CT became a focus at the national level in the United States.

Around the world, computer science and CT education is now expanding and being increasingly recognised within formal K-12 education settings. In a 2020 report, the Brookings Institute surveyed 219 countries to identify which had online evidence of in-school computer science education in place in K-12 schools (Fowler and Vegas, 2021[39]). The report found that 44 countries mandated that schools offer it as an elective or required course; that 15 countries offered computer science in select schools and some subnational jurisdictions; and that 160 countries were only piloting computer science education programmes or had no available evidence of in-school computer science education, the curriculum being particularly rare in low-income countries.

Today, within the OECD, Israel, New Zealand, and South Korea have all included computer science in their national secondary education curricula, and several others are following. However, progress in this respect has been more limited when it comes to primary school and the early education years. The following section breaks down the current state of CT frameworks and standards across OECD countries, along with major initiatives and organisations that are leading the way with CT in early education.

3.2. Exploring current CT initiatives and frameworks across OECD countries.

As computing has grown increasingly important in today's world, the public demand for education that supports CT and computer science is high (K-12 Computer Science Framework Steering Committee, 2016[40]). Most parents report wanting their child's school to offer computer science (Google/Gallup, 2015[41]). In meeting these needs, a growing number of OECD countries have taken steps in recent years to incorporate some form of computer science or CT into their curricular frameworks. Countries like Australia, Canada, Chile, South Africa, Korea and the United Kingdom all have computer science present in educational frameworks or guidelines for primary school or earlier. The emergence of curricular standards and frameworks in OECD countries generally emerged through collaborations between local and national governments, technology industry leaders, educators, and researchers. Examples of major initiatives in support of CT and computer science include:

- The **K-12 Computer Science Framework (United States)**: Created by educators, local government, and the largest technology companies, used as a model in the United States and many other countries. The framework was developed to inform the development of standards and curriculum, build capacity for teaching computer science, and implement computer science pathways (K-12 Computer Science Framework Steering Committee, 2016[40]).
- **CS For All (United States)**: A central resource for individuals and organisations interested in K-12 computer science education in the United States. The initiative includes policy work at the local, state, and national levels, school and district innovation, teaching, and research.
- **Informatics for All (Europe)**: A coalition that promotes the inclusion of computer science and informatics in schools across European countries. The Informatics for All coalition was formed in 2018 by the joint efforts of the ACM Europe Council, the CEPIS Education Committee, and Informatics Europe. These organisations share a common concern about the state of informatics education throughout Europe, and are committed to promoting activities that will improve it. The Informatics for All initiative deploys a two-tier strategy at all educational levels: informatics as an area of specialisation that is, as a fundamental and independent subject in school; and the integration of informatics with other school subjects, as well as with study programmes in higher education.
- **Computing at School (United Kingdom)**: A non-profit organisation which established a coalition of industry representatives, teachers, and parents in 2008. The organisation went on to play a pivotal role in rebranding the information and communications technology (ICT) programme of study in 2014 into a computing programme that placed a greater emphasis on computer science (The Royal Society, 2021[42]). By changing the programme, the government instructed schools to provide more rigorous instruction in computer science concepts like Boolean logic and programming languages.
- **National Centre for Computing Education (NCCE) (United Kingdom)**: In 2018, Parliament and the Department for Education allocated 84 million pounds to establish the National Centre for Computing Education (NCCE) to train teachers (Cellan-Jones, 2019[43]). Drawing on help from non-profit organisation partners, the Centre creates lesson plans and resources, runs training programmes, and offers certification for pre-service and in-service teachers. Since its opening, the Centre has engaged 29 500 teachers in training, 7 600 of which have benefited from continuous professional development (Fowler and Vegas, 2021[39]).

In most countries, the main rationale for introducing CT and coding is to foster 21st century skills, which are seen as essential for active participation and employment in the increasingly digital job market. Approaches to doing so, however, vary distinctly from country to country. For example: Austria, Denmark and Hungary focus mainly on logical thinking and problem solving processes as learning outcomes. Finland and Turkey implement both process-based learning goals (e.g. logical thinking and problem solving) as well as skill-based learning goals specific to coding in their frameworks (New South Wales Department of Education, 2019[38]).

In the OECD area specifically, three countries serve as case examples for their approaches to computer science education in early childhood and the primary school years. In primary education, Estonia has a national cross-curricular theme called ‘Technology and Innovation’ which requires all teachers to implement technology in their teaching. In Korea, the Software Education programme focuses on developing CT, coding skills and

creative expression through software at all levels of education. Primary and lower secondary schools were included in this focus as of 2018. Finally, the United Kingdom introduced in 2013 a rebranded "computing" curriculum which prioritised computer science concepts, as opposed to the computer literacy topics that were emphasised before the change. The UK national curriculum for computing aims to ensure that all pupils can understand and apply the fundamental principles and concepts of computer science, including abstraction, logic, algorithms and data representation.

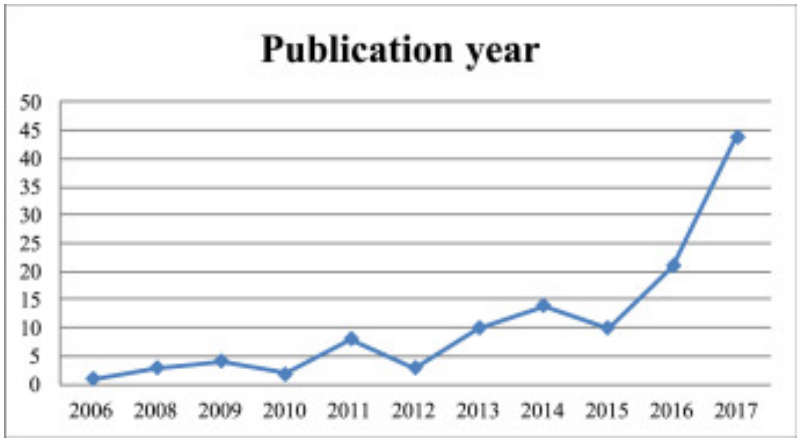
Having state/province and countrywide frameworks for teaching CT and computer science in early childhood education can increase children's exposure to CT. In a recent paper analysing quantitative non-identifying data from Google Analytics on users of the popular ScratchJr programming application in the United States, results show that states with computer science standards had more ScratchJr users on average and had more total sessions with the app on average (Sullivan and Bers, 2019[44]). Results also show preliminary evidence that states with computer science standards in place have longer average session duration as well as a higher average number of users returning to edit an existing project (Sullivan and Bers, 2019[44]).

3.3. Recent international research on CT in early education

With the rising prevalence of CT initiatives and curricula, there has also been an increase in research on CT in education, especially after Wing's (2006[1]) seminal article proposing CT as a critical foundational academic skill. Since then, various studies have set out to define CT and its effective implementation in education. A recent meta-analysis by (Hsu, Chang and Hung, 2018[45]) categorised 120 academic articles and books published between 2006 and 2016 (mostly covering education beyond pre-primary), to arrive at evidence-based CT teaching strategies (see Figure 2). Further, the study identified 59 different definitions based on several concepts such as problem solving, technology, thinking, individual and social qualities, and further noted that general statements on "thinking" prior to 2006 were replaced by statements with a focus on problem solving and technology.

The meta-analysis found that the main successful teaching strategies for primary education and below included scaffolding (adults offering support for learners during activities), universal design for learning (a design framework to provide flexible learning environments and interfaces), project-based learning (a student-centred pedagogy that involves dynamic exploration of real-world challenges and problems), and problem-based learning (another student-driven pedagogy that involves pursuing solutions to open-ended problems). Researchers are also exploring best practices and benefits for bringing CT to early childhood. Results from a two-year longitudinal study in Canada on integrating tablet (e.g. iPad) equipment in 14 kindergarten classrooms confirms research in other countries, showing that these tools afforded children the ability to create multimodal productions that were longer, more complex, and more varied than their literacy production with traditional literacy tools and practices (McGlynn-Stewart et al., 2019[46]). At a policy level, the development of numeracy performance standards in the British Columbia curriculum provides an example of CT by taking a project/problem-based learning approach to assessment, encouraging teachers to allow students to develop and demonstrate their numerate thinking and communication skills (Interpret, Plan, Solve, Analyse, Communicate) through open-ended problems in various learning areas.

Figure 2.Number of new global public academic journal articles on computational thinking (2006-2017).



Source: Hsu, Change, and Hung, (2018[45]), "How to learn and how to teach computational thinking: Suggestions based on a review of the literature" *Computers & Education*, 126, 296-310, https://doi.org/10.1016/j.compedu.2018.07.004.

4. Computational thinking and early learning and development

4.1. Support and criticism of CT in early education

CT advocates claim wide-reaching benefits of its integration in mainstream education, primarily citing cognitive skill and competence building, creative expression and broadened participation, as well as applications for social justice and ethics development (Kafai, Proctor and Lui, 2020[47]). Relevant to early childhood, many claims rely on theories of CT skills mapping onto other cognitive and socio-emotional learning domains such as literacy, numeracy, general problem solving, and more. Empirical research is presented in the following sections to identify trends in CT education in various domains, to highlight potential opportunities and challenges within this burgeoning field. However, it is important to note that this body of research is in its infancy and some claims are yet lacking systematic verification in empirical studies. Furthermore, experiences of integration of CT in pre-primary education at a large scale (e.g. at country or subnational levels) are rare or very recent, which makes it difficult to assess their outcomes conclusively. Robust research designs, including randomised controlled trials, have yet to generalise in this field in order to generate more conclusive evidence on the potential causal links between CT skills and different dimensions of early cognitive and socio-emotional development, as well as on the impact of CT education interventions. With these caveats in mind, it is nonetheless worth reviewing the existing evidence to identify both promising aspects and limitations of some of the initiatives addressing this area of early digital literacy. Where most researchers agree is that, in addition to technological tool and learning domain, the educational context, instructional format, and intended pedagogy are all important when evaluating benefits and constraints of early CT initiatives.

4.2. Exploring the role of CT in early learning and development

In 2006, Jeannette Wing proposed that acquiring CT skills can have benefits for the development of other domains of thinking (Wing, 2006[1]). She posited that CT includes thought processes such as thinking abstractly and using efficient problem solving strategies

in order to figure out the solution to complicated tasks and problems when figuring out a complex task and using heuristic reasoning to discover a solution (Wing, 2006[1]; Wing, 2011[2]). Mastering CT includes the processes of pattern recognition, conceptualisation, planning, and problem solving. Therefore, CT skills are not only valuable for computer programming but helpful in a variety of other contexts such as solving mathematical problems, planning and organising for a large event, sequencing a storyboard, and more (Román-González, Moreno-León and Robles, 2019[48]; Zhang and Nouri, 2019[49]).

Compared to research on computer science and associated tools (e.g. robotics kits, programming languages, apps, and games) with older children and adults, little is still known about the connections between computer science and early learning. In recent years, there has been a slowly growing body of work examining how computer science tools can be used to foster CT as well as young children's rapidly developing cognitive skills and executive functions. The following sections look at this newer research and explore how CT has been shown to impact cognitive and social-emotional skills during the foundational early childhood years.

4.3. CT and cognitive development

4.3.1. Cognitive skills and executive functions

Early studies with the text-based programming language Logo were among the first studies to demonstrate that computer programming can help young children with number sense, language skills, and visual memory (Clements, 1999[50]). A meta-analysis of 65 studies revealed that students who participated in computer programming typically scored higher on various cognitive-ability assessments than children who did not participate.

More recently, research has evaluated the graphical programming language ScratchJr which is designed for children aged five to seven and reported to have been used in all but five countries in the world (Bers, 2018[9]; Leidl, Bers and Mihm, 2017[51]; Sullivan and Bers, 2019[44]). Studies have found that young children are able to use ScratchJr to create personally meaningful projects and demonstrate CT and problem solving strategies, and that these experiences are especially successful when educators allow children to explore and engage in child-led free-play (Bers, 2018[9]; Bers, 2020[36]; Leidl, Bers and Mihm, 2017[51]; Portelance, Strawhacker and Bers, 2015[52]; Strawhacker, Lee and Bers, 2017[53]; Strawhacker et al., 2015[54]). However, more research is needed to determine if the observed associations in these early-stage studies can be causally linked to coding interventions directly. For example: a controlled experimental trial of 28 children aged 4-5 years found that after a coding classroom experience, children in the experimental group showed an increase in non-verbal cognitive abilities, but there was no statistically significant difference in their problem solving skills (Çiftci and Bildiren, 2019[55]). Relatedly, a study of 49 primary students (aged 10-11 years) who engaged in a coding course with the Scratch environment found no significant differences in the problem solving skills of the students after the intervention; instead, they found a non-significant increase in students' ratings of self-confidence in their own problem solving ability (Kalelioglu and Gulbahar, 2014[56]). This suggests that future work might unpack the role of screen-based CT motivation, engagement, and self-concept as they relate to children's problem solving abilities.

Behaviourally, children using ScratchJr are encouraged to engage in the engineering design cycle to create their projects, a critical aspect of CT, as well as to leverage math concepts of cardinality, sequencing, and order, and foundational literacy practices of drafting, revising, and sharing story compositions, exploring and utilising narrative structures, and decoding symbols (Bers, 2020[36]) (Flannery et al., 2013[57]) (Hassenfeld and Bers,

2020[58]). A 2014 study of 98 K-2 students and 9 teachers from 2 schools revealed that coding with ScratchJr supported symbol recognition and sequencing skills at an appropriate developmental level for average K-2 children (Strawhacker and Bers, 2014[59]), and that different programming blocks (e.g. conditional statements, repeat loops) supported increasing levels of programmatic complexity, allowing the app to "grow with the child" from kindergarten through second grade (Portelance, Strawhacker and Bers, 2015[52]). Additionally, a series of studies conducted with preschoolers and kindergarteners showed that coding can significantly improve young children's sequencing ability, an important pre-math and pre-literacy skill, on a standardised picture sequencing assessment unrelated to programming (Kazakoff and Bers, 2014[60]) (Kazakoff, Sullivan and Bers, 2012[61]). Research on computer programming and tangible robotics construction sets have also shown connections to cognitive development (Flannery et al., 2013[57]) (Strawhacker, Lee and Bers, 2017[53]). For example, prior research has demonstrated that robotics can help children develop a stronger understanding of mathematical concepts such as number, size, and shape in much the same way that traditional materials like pattern blocks, beads, and balls do (Resnick, 1998[62]) (Brosterman, 1997[63]). Other research has shown educational computer programming as a medium to develop foundational skills of math, logic, and sequential ordering (Kazakoff, Sullivan and Bers, 2012[61]) (Kazakoff and Bers, 2014[60]) (Pea and Kurland, 1984[64]).

Research with the KIBO robotics kit with children ranging from ages 4-8 has demonstrated that young children can practice important cognitive skills of problem solving and debugging when engaging with coding the robot (Sullivan, Bers and Mihm, 2017[65]) (Sullivan, Elkin and Bers, 2015[66]). More recent research with children ages 7-9 using KIBO indicated that learning to code with robotics improves young children's problem solving skills, particularly in children who generalise the knowledge gained from coding into broader CT skills (Relkin and Bers, 2020[25]).

CT and computer programming can also help young children practice their developing executive function abilities, which consist of mental flexibility, inhibitory control, and working memory (Center on the Developing Child at Harvard University, 2011[67]) (Blair and Diamond, 2008[68]). CT, as a means of problem solving, taps into similar and overlapping cognitive functions, many of which are considered under the umbrella of executive function, and, by extension, self-regulation (Myers, 2021[69]). For example, when using the ScratchJr programming language, children must draw on their working memory to remember their given programming challenge, remember the programming blocks that correspond to the actions they want their characters to take, and remember the syntax rules inherent to this language (Kazakoff and Bers, 2014[60]).

4.4. CT and social-emotional development

Early childhood is a critical developmental period for learning necessary social skills through peer-to-peer interactions that help develop social knowledge of the peer group and differentiate friends from playmates (Hartup, 1983[70]; Howes, 1987[71]). For young children who are just beginning to learn how to collaborate and work together with peers, the design features of many computing technologies can be used to promote social and pro-social development (Bers, 2021[3]) (Bers, 2022[72]). For example, tools that are designed to allow multiple children to work together on one project (e.g. robotics kits that allow one child to construct the robot while another child programmes the robot) and digital tools that allow for sharing and "re-mixing" (e.g. programming applications with a "share" or sending feature) can foster collaboration and social development in ways that other tools cannot. Early work with technology and young children has shown that computers can serve as catalysts for social interaction in early childhood education classrooms (Clements,

1999[50]), and an experience with primary education children has shown that children can have twice as many social interactions in front of a computer than when they are doing other activities (Svensson, 2000[73]). Programming in groups invites children not only to collaborate but also to engage in social play and to develop the social coordination skills and social scripts that are necessary for negotiating, problem solving, sharing, and working within groups (Erickson, 1985[74]; Pellegrini and Smith, 1998[75]; McElwain and Volling, 2005[76]).

Children are also more likely to ask their peers for help when using a computer, even when an adult is present, thereby increasing the amount of peer collaboration in the classroom (Wartella and Jennings, 2000[77]). Other research has shown that it is not just the technology itself that serves as a catalyst to collaboration, but also the way the technology is implemented in curricula and classroom activities. For example, research on collaboration and the use of programmable robotics kits for young children has shown that using a more unstructured pedagogy that embodies a "learn by doing" approach serves to foster more peer collaboration than a more structured teaching approach with more teacher guidance (Lee, Sullivan and Bers, 2013[78]).

Research in home settings has been sparse and less conclusive about the positive impact of digital experiences for children's socio-emotional learning. A study on a nationally representative sample of 4 914 children aged 0–5 in Germany assessed children's socio-emotional, practical life skills, and academic competencies via a standardised parental survey, and compared those who reported a greater access and frequency of digital tools in the home learning environment to those with relatively more analogue home environments (Lehrl et al., 2021[79]). The study concluded that for preschoolers, digital home learning activities were associated with weaker socio-emotional skills but higher academic skills. Importantly, this study does not differentiate between digital access and CT engagement, which may point to a broader finding noted in other literature reviews about challenges in parental uptake and understanding of how to effectively scaffold digital experiences for children's learning (Wan, Jiang and Zhan, 2020[80]).

4.5. CT and the positive technological development framework

The previous section described the ways that new computing technologies can be used to foster a range of socio-emotional skills. Along these lines, the Positive Technological Development (PTD) framework developed by Bers provides a model to guide the development, implementation and evaluation of educational programmes that use new technologies to promote learning as an aspect of positive youth development (Bers, 2012[81]) (Bers, 2020[36]). The PTD framework is a natural extension of the computer literacy and the technological fluency movements that have influenced the world of education but adds psychosocial and ethical components to the cognitive dimension. As a theoretical framework, PTD proposes six positive behaviours (six C's) that should be supported by educational programmes that use new educational technologies, including but not limited to, tools that support CT. These positive behaviours are: content creation, creativity, communication, collaboration, community building, and choices of conduct (Bers, 2012[81]) (Bers, 2008[82]) (Strawhacker, Lee and Bers, 2017[53]) (Lee, Sullivan and Bers, 2013[78]). Some of these pertain to behaviours that enrich the intrapersonal domain (content creation, creativity, and choices of conduct); others address the interpersonal domain and look at social aspects (communication, collaboration, and community building).

In the context of CT, the PTD framework provides practical recommendations to integrate technology with meaningful learning goals in mind, including but not limited to CT concepts such as debugging and sequencing, by engaging children in inter- and

intrapersonal play with technologies. This foundational research on technology-based pedagogical approaches has inspired more recent approaches emphasising CT specifically (Kong, 2016[83]; Tsortanidou, Daradoumis and Barberá, 2021[84]), which have similarly taken up the perspective that CT comprises universal skills that apply broadly beyond specific computer science fields.

4.6. Integrating across STEAM curricula

In the growing international discussion around STEM education, how to effectively teach technology and engineering has become more pressing to researchers and educators (Granovskiy, 2018[85]; US Government National Science and Technology Council, 2018[86]; Department for Education, 2013[37]). Research confirms that an integrated approach to STEM education, in which activities cut across several disciplines, is developmentally suited for early childhood contexts (Aldemir and Kermani, 2016[87]; Wortham, 2009[88]). Historically, early childhood STEM education has focused on foundational numeracy skills and natural sciences awareness (Bers, 2008[82]; Bers, Seddighin and Sullivan, 2013[89]; Moomaw and Davis, 2010[90]). The idea of promoting creativity and expression through STEM is articulated in a newer acronym called "STEAM" (Science, Technology, Engineering, *Arts*, Mathematics) that is growing in popularity across the United States and worldwide (Allen-Handy et al., 2020[91]; Watson, 2020[92]; Yakman, 2008[93]). The "A" of STEAM represents the whole spectrum of the liberal arts, including language arts, social studies, music, visual arts, and more.

Within an early childhood context, STEAM education means finding ways for children to explore these subjects in an integrated way through hands-on projects, books, discussions, experiments, art explorations, collaboration, games, physical play, and more (Sullivan and Strawhacker, 2021[94]). New technological tools such as robotics kits and programming languages designed for young children have become a popular way to teach interdisciplinary STEAM content, as they allow for an integration of arts and crafts, literacy, music, and more with engineering and robotics (Barnes et al., 2017[95]) (Bravo Sánchez, González Correal and González Guerrero, 2017[96]) (Elkin, Sullivan and Bers, 2016[97]) (Sullivan, Strawhacker and Bers, 2017[98]). For example, the *Dances from Around the World Curriculum*, is a robotics and programming curriculum that promotes an integration of technology and engineering concepts with an exploration of music and culture, engaging children to build, code, and share a robot representation of a personally meaningful music and dance performance (Sullivan and Bers, 2017[99]).

A primary motivation for introducing CT practices into activities targeting other areas of the curriculum across a STEAM framework is the rapidly changing nature of many disciplines as they are practiced in the professional world (Bailey and Borwein, 2011[100]) (Blikstein and Wilensky, 2009[101]) (Hambrusch et al., 2009[102]) (Henderson, 2007[103]) (Rubin and Nemirovsky, 1991[104]) (Sengupta et al., 2013[105]) (Settle et al., 2012[106]) (Settle, Goldberg and Barr, 2013[107]). For example, in the last 20 years, nearly every field related to science and mathematics has seen the growth of a computational counterpart, such as Bioinformatics, Computational Statistics, Chemometrics, and Neuroinformatics (Weintrop et al., 2015[108]). From a pedagogical perspective, the authenticity and real-world applicability of integrating computer science and other disciplines is important in the effort to motivate diverse and meaningful participation in activities that require computational, mathematical, scientific, and linguistic thinking (Blikstein, 2013[109]; Chinn and Malhotra, 2002[110]; Confrey, 1994[111]; Fisher and Margolis, 2003[112]; Margolis, 2017[113]; Ryoo et al., 2013[114]).

4.6.1. Literacy

Computer programming became associated with the technology (T) dimension of STEM when it first emerged in early childhood education (Ryoo et al., 2013[114]) (Clements, 1999[50]) (Guzdial and Morrison, 2016[115]). However, this categorisation was rooted in the assumption that disciplines complementary to programming are math and science. For example, a recent meta-analysis of 105 empirical studies looked at transfer of computer programming skills to other cognitive domains and found positive transfer to situations that require creative thinking, mathematical skills and metacognition, but very little transfer effect to students' literacy skills (Scherer, Siddiq and Sánchez Viveros, 2019[116]). The authors concluded that reading comprehension and writing skills must overlap only marginally with programming skills, but CT advocates argue that this could be an issue of implementation and ingrained traditions in CT instruction. The assumption that coding is primarily a math and science skill has led to the creation of robotics and computer science applications that are based on solving challenges with increased complexity and leave out the creative and self-expressive aspects of programming that align more closely with literacy, such as telling a story, conveying ideas, and expressing creativity (Hassenfeld et al., 2020[117]). This has limited the exploration of using computer programming explicitly to foster and support literacy and the arts. Researchers interested in CT and literacy integration take a different position and argue that there is significant overlap when using natural and artificial languages (Fedorenko et al., 2019[118]) and there may also be theoretical overlap between writing skills and programming skills (Bers, 2019[119]; Vee, 2013[120]; Vee, 2017[121]). More recently, programming has been integrated with the development of language and literacy to fill these gaps (Aguirre-Muñoz and Pantoya, 2016[122]) (Maguth, 2012[123]) (Sullivan and Bers, 2017[99]) (Sullivan, Strawhacker and Bers, 2017[98]) (Bers, 2019[119]).

A recent initiative in the United States called "Coding as Another Language" explores the ways in which the process of teaching computer science to young children can resemble the educational process used for teaching literacy and seeks to identify the overlapping associated cognitive and socio-cultural mechanisms. The "Coding as Another Language" project (Bers, 2019[119]) (Bers, 2019[124]), involves several dimensions: 1) the creation of programming environments explicitly designed with a literacy approach, 2) resources, such as the free CAL (Coding as Another Language) curriculum for ScratchJr and KIBO, which present the process of coding as a semiotic act, a meaning making activity, and not only a problem solving challenge, 3) a theoretical framework (Bers, 2020[36]), 4) a pedagogical approach with professional development strategies that explicitly highlight the connection between the activity of coding and the mastering of a language and its uses to convey meaning, 5) research studies in classrooms to understand the affordances of this approach compared to others, and 6) experimental studies in lab settings to characterise cognitive mechanisms using fMRI and other neuro imaging techniques to explore the relationships between language networks in the brain and computer programming.

The vision of "coding as literacy" is growing (Vee, 2017[121]). In 2021, researchers and practitioners in the fields of computer science, language and literacy, and STEM education developed a shared vision of the conceptual relationship of computing to language and literacy development and of evidence-based perspectives on how to support multilingual students in learning computer science (Jacob, Parker and Warschauer, 2021[125]). This includes a theoretical model that distinguishes between CT *as* literacy, *through* literacy, and literacy through CT.

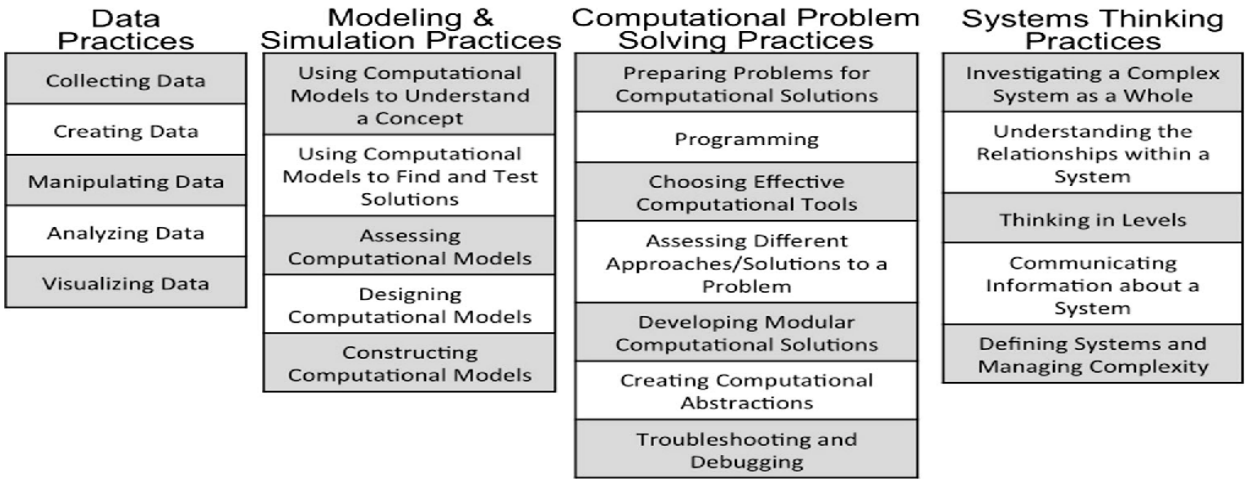
4.6.2. Mathematics

As technology plays a growing role in the lives of children, the long-term trajectory of mathematical literacy should also encompass the synergistic and reciprocal relationship between CT and mathematical thinking. Some researchers have argued that CT supports mathematics in many ways, but particularly because mathematical reasoning complements the problem solving skills that encompass CT (Gadanidis, 2015[126]; Rambally, 2017[127]). In a Canadian research study with kindergarten teachers, teachers reported that they found a considerable overlap between CT and mathematical thinking in activities engaged in by the children (Kotsopoulos et al., 2019[128])

Computer science has several parallels with mathematics. For example, scholars have argued that the skill of abstraction, and intentionally moving among different levels of abstraction (e.g. attending to a real-world phenomenon and a simulated model of that phenomenon), is a critically important skill for both computer scientists and mathematicians (Kramer, 2007[129]; Hazzan, 2008[130]; Rich, Yadav and Schwarz, 2019[131]). A recent analysis of the Common Core Math Curriculum (CCSS-M) for grades K-5 in the United States suggested that elementary mathematics concepts offer opportunities to begin a spiral curriculum, emphasising CT ideas in early elementary mathematics, to be then expanded in computer science contexts in later grades (Bruner, 2009[132]; Rich et al., 2019[133]).

Although CT has been included as a core practice in mathematics' standards, the current questions facing CT in mathematics education focus on implementation. Weintrop et al. (2015[108]) proposed a definition of CT for mathematics and science in the form of a taxonomy consisting of four main categories: data practices, modelling and simulation practices, computational problem solving practices, and systems thinking practices (Figure 3). This contribution represents an attempt to converge on concise and specific learning concepts to further develop standards, curricula, and assessments.

Figure 3. Computational thinking in mathematics and science taxonomy



Source: (Weintrop et al., 2015[108])

4.6.3. Science

Computer science can bring creative agency and hands-on exploration to science lessons, which can be abstract and overly structured for young children. Introducing novel scientific

topics brings real-world relevance and context to STEM explorations, connecting children's learning to topics in their broader community and society. New technologies offer children a chance to playfully explore natural organisms and phenomena, some of which can otherwise be too microscopic, invisible, or time-consuming to explore in a meaningful way, and even to apply concepts about software and sequencing learned from their favourite coding toys to advanced ideas like gene sequencing (Strawhacker et al., 2020[134]). For instance, the tangible CRISPEE technology, a prototype developed by the DevTech Research Group at Tufts University and the Wellesley College Human Computer Interaction Lab in the United States, engages children in coding with a tangible block-based language to computationally design bioluminescent animals (like fireflies) to glow in certain colours, and to change colour depending on environmental indicators (Strawhacker et al., 2020[135]; Strawhacker et al., 2020[136]; Strawhacker et al., 2020[137]).

There are also overlapping themes that young children explore in early childhood that are foundational to both CT and scientific reasoning. Science and computer science share similar methodologies for asking and answering questions (i.e. the scientific method) and for building and testing solutions to human problems (i.e. the design process). Both involve processes that young children practice starting in kindergarten, such as ideating/imagining, designing (experiments or prototypes), iterating, and refining. Similarly, computer science and life sciences like biology both rely on computational concepts such as abstraction, modularity, and algorithmic logic to understand and model how structures (e.g. of organs and cells, or codes and functions) operate within hierarchies to function as a system. These computational concepts may sound highly sophisticated, but educational coding tools like the ScratchJr programming language, KIBO robotics kit, BeeBot robot, Code-a-Pillar, aim to introduce those concepts to children as early as preschool.

5. Tools for early CT learning

Researchers are addressing the implications of young children's exposure to digital technology. Because so many of the tools and technologies that support CT include screen time and/or Internet access, it is important to highlight research and recommendations on safe and developmentally appropriate practice using digital technology in early childhood.

5.1. Designing technologies for CT learning

Research has shown that many of children's best learning experiences come when they are engaged not simply in interacting with materials but in designing, creating, and inventing with them (Folk, 1981[138]; Resnick, 2002[139]; Resnick, 2006[140]). Education resources like the Youth Maker Playbook (Davee et al., 2015[141]) espouse the importance of inspiring children to become producers of their own creative, playful, and functional artefacts, rather than simply consumers of other people's work. Not all technologies are created equal, and many described later in this review are specifically designed to empower children to be the directors of their own playful and creative experiences.

5.1.1. Developmentally appropriate design and opportunities for play

Programming languages, as with natural languages, can be used for a variety of purposes, from mundane and repetitive to creative tasks. The intention of the user of the language determines how much creativity is displayed. The language is a vehicle, a medium, for expression. Young children, with their own developmental needs and abilities, need programming languages specifically designed for them. These must be simple languages that still support multiple combinations, have syntax and a grammar, and offer multiple

ways to solve problems – and thus, be more like playgrounds rather than playpens (Bers, 2018[9]). For example, a "playground" programming environment will allow open-ended opportunities for a child to create, explore, experience failures, and encounter challenges. Meanwhile, a "playpen" programming environment will be more restricted and adult-directed and might only engage children in specific drilling of concepts and follow a series of sequential levels instead of letting the child drive the experience. Technologies also need to provide opportunities for creating a computational artefact that can be shared with others and support a growing range of computational literacy skills, from beginners to experts. Children who are fluent in a particular programming language are more likely to learn a second one with ease, and more likely to have mastered some aspects of CT and to be able to transfer that mastery to different situations.

When designing (or choosing) developmentally appropriate programming environments for young children, certain design features should therefore be carefully considered. For very young children who are not yet reading independently, programming tools should offer visual (i.e. picture, symbol, or icon based) languages as opposed to text-based languages. The programming languages should offer a syntax and grammar that can be mastered to create scripts of multiple levels of complexity and they should support multiple combinations and solutions (as opposed to supporting just "one correct outcome"). In English-speaking countries, it may also be important to consider tools that allow programming scripts to run as a sequence from left to right instead of the traditional top-to-bottom format of most adult programming languages, to reinforce print-awareness and English literacy skills (Flannery et al., 2013[57]). Perhaps most importantly, (Resnick et al., 2009[142]) proposed that programming environments should have what Seymour Papert described as "low floors, high ceilings, and wide walls". This means that learners should be able to create something easily right away (low floors), maintain their interest over time as they create progressively more complex projects (high ceilings), and allow students across a multitude of learning styles, cultures, and interests to learn and develop (wide walls). In this way, a programming tool can grow with children as they develop their skills and broaden their experience (Portelance, Strawhacker and Bers, 2015[52]).

Programming tools should also be designed with a playful approach when thinking about early childhood education (Bers, 2020[36]). Research in early childhood has shown that play is a wonderful way for children to learn (Garvey, 1977[143]) (Fromberg and Williams, 1992[144]). Play has been described as a vehicle for the development of imagination and intelligence, language, social skills, and perceptual motor abilities in young children (Frost, 1992[145]). Play enhances language development, social competence, creativity, imagination, and thinking skills and has been described as the "ultimate integrator of human experience" (Fromberg, 1990, p. 223[146]). When children play, they draw upon their past experiences, including things they have done, seen others do, read about, watched on television, or seen through other media. They integrate these experiences into their games and play scenarios, and they express and communicate their fears and feelings.

When programming is taught with a playful approach, children are not afraid to make mistakes. Pretend play in early childhood enhances the child's capacity for cognitive flexibility and, ultimately, creativity (Russ, 2003[147]; Singer and Singer, 2005[148]). Csikszentmihalyi (1981[149]) describes play as "a subset of life… an arrangement in which one can practice behaviour without dreading its consequences". Programming with a playground approach offers similar opportunities. It looks different from traditional computer science courses in which students need to solve a challenge under time pressure or find the pre-determined correct way to answer a prompt.

5.1.2. Digital games and puzzle-style software applications

There is a growing number of digital games and puzzle-style software applications aimed at supporting young children's learning of computer science concepts and CT skills, without the need of experimenting with a programming language.

Most of these focus on sequencing and logic as they engage children in progressing through problem solving levels in a typical game-like fashion. For example, the game Lightbot is a popular programming puzzle game for young children, in which the goal is to complete pre-set tasks such as making a robot light up all of the blue tiles on a 3D grid. Children programme their screen-based robot with a series of instructions. There are different versions of Lightbot for different ages, including Lightbot Jr. for young children ages 4-8.

Other popular programming games include Kodable, which includes maze-like levels, and Cargo Bot, which engages young children in learning programming concepts while using a crane to move boxes back and forth between platforms. The website Code.org offers a variety of coding games for children, ranging in age from young children (categorised as "pre-readers") up through high school, as well as Hour of Code activities including Candy Quest (a multi-level coding quest for candy), Code with Anna and Elsa (explore coding with characters from the popular movie *Frozen* by helping them create snowflakes and more), Dragon Blast (embark on a quest for treasure using coding skills), and more. In Code.org's "Classic Maze game" kids write lines of code in a setting inspired by the popular game Angry Birds. In this game, players help Angry Birds get to the Naughty Pigs. Each level becomes increasingly difficult to navigate and focuses on different coding concepts. Programming games as in these examples tend to appeal to young children who enjoy a style of play akin to video games, with specific levels to beat and sequential tasks to complete.

Research on students using Code.org's "Classic Maze" activity and the "Flappy Code" activity found that students showed significant changes in their attitudes towards and self-efficacy with computer science after engaging in just one Hour of Code activity (Phillips and Brooks, 2017[150]). However, it is important to note that these games present a more limited set of experiences as compared to block-based programming languages. While programming *languages* offer an open-ended setting to create any project of choice, thus providing more opportunities for creative experiences, programming *games* are typically more limiting and prompt players to explore and practice a particular aspect of programming such as cause and effect, sequence, logic and problem solving (Sullivan and Bers, 2019[44]).

5.2. Open-ended coding and programming environments

Many programming interfaces for children offer simple map-based puzzles with gamified elements (e.g. move a character along a path and avoid roadblocks to earn a gold star). These tend to use programming mainly as a directional steering technique, sometimes within a story context, and they offer step-by-step instructions and guiding prompts (e.g. Cato's Hike, Code Monkey Island, Code.org's Code Studio, Daisy the Dinosaur, LittleCodr, Nancy Drew: Codes and Clues, Robot Turtles, Tynker). Other programming environments take a more haptic approach, either by programming a physical robot, or by using gestures and physical movements in the programming experience.

Open-ended coding and programming environments offer the most playful learning opportunities. They can be tangible, screen-based, or a combination, although evidence suggests that tangible tools may be more effective as a first introduction to programming in the early years (Manches and Price, 2011[151]; Pugnali, Sullivan and Umashi Bers,

2017[152]). At early ages, there is a pedagogical incentive to use block-based programming, a hands-on way to encourage the child to explore coding, as well as to use languages with simple movement commands like forward, side or back, to support spatial, vision, and cognitive skills (Silva, Dembogurski and Semaan, 2021[153]; Bers, 2020[36]).

ScratchJr is a digital playground for coding, designed by a team of researchers and developers at Tufts University' DevTech Research Group, the MIT Media Lab's Lifelong Kindergarten Group, and the Playful Invention Company, specifically created to invite playful exploration (Bers and Resnick, 2015[154]; Flannery et al., 2013[57]). Children can design characters and backgrounds and snap together graphical programming blocks to make their characters move, jump, dance, and sing. They can modify characters in the paint editor, create colourful backgrounds, add their own voices and sounds, and take photos of themselves to insert into their stories, games, or animated collages.

Osmo is another digital playground-style programming environment, designed (under the prototype name "Strawbies") by the TIDAL Lab at Northwestern University (Hu et al., 2015[155]). Osmo integrates a screen-based interface with tangible programming tiles, creating an experience that draws children into collaborative play. The use of tangibles increases the visibility of game play, allowing it to move beyond the screen and spill out into the real world.

MaKey is a kit that lets children transform everyday objects into computer interfaces. From make a game pad out of Play-Doh, a musical instrument out of bananas, or creative inventions. It consists of a USB device than be plugged into a personal computer and used to make personal switches that act like keys on the keyboard. This is where the name originates: Make + Key = MaKey! As a "plug and play" device, it does not require any electronics or programming skills and is automatically compatible with any existing software users wish to use, including visual programming environments designed for children.

Another kit called littleBits offers easy-to-use electronic building blocks that snap together with magnets. The goal of littleBits is to make learning about circuitry and electronics exciting and engaging for children and adults alike. The parts of the kit can connect together to create complex circuits in seconds. In addition to simple outputs like lights, speakers, and motors, littleBits parts include light sensors and pressure sensors, and many switches, dimmers, and other ways to control the circuit current, allowing for advanced electrical engineering explorations that can be coded through an on-screen app. The complex circuitry that the kit affords, and the fact that circuit parts are small enough to be a choking hazard for young children, makes the kit developmentally appropriate for its recommended age range of 8+ years.

Calliope mini also offers a new and tangible way to explore coding. Similar to Makey, Calliope mini is a microcontroller with a screen-based coding interface that allows children to create simple programmes using many different inputs (e.g. buttons, switches), outputs (e.g. lights, speakers), and sensors (including unique data-based sensors like a compass, radio, and Bluetooth reader). Children can for instance explore e-textiles by making interactive clothing, or create traditional wired circuits, but the kit requires a high level of fine motor skills to access tiny connection ports. This kit is marketed for children ages 8+, the high end of the age range of interest in this review.

5.3. Media (TV) for computational thinking

There are a growing number of television shows focused on introducing CT skills, computer programming concepts, and computer science more generally. In the United States, the Corporation for Public Broadcasting (CPB) and PBS received a Ready To Learn

grant from the US Department of Education's Office of Elementary and Secondary Education to fund a comprehensive multi-media learning and station engagement initiative (Public Broadcasting Service, 2020[156]). The initiative will result in the development of new content to help young children build skills to help them succeed in school and life, including CT, and show them career options in age-appropriate ways. For example, CPB and PBS are working with experts in early learning and leading children's media producers to create two shows, called *Wombats!* and *Liza Loops*, that will integrate CT learning alongside critical thinking, collaboration and other skills.

Many other popular television programmes have emerged in recent years in the United States that reference or attempt to explicitly teach concepts about computers, computer science, programming, and CT. These include *Blaze and the Monster Machines*, *Annedroids*, *Storybots*, and more. Many of these new shows build on earlier work done by the British Broadcasting Corporation (BBC) in 2014, when the country's national computing curriculum first emerged. The BBC's technology-themed TV shows first broadcasted around 2014 included outputs like *Technobabble*, an app and gadget-themed show designed to encourage its audience to expand its computer skills; *Appsolute Genius*, with interviews to prominent computer programmers, including the creators of Sonic the Hedgehog and Pac-Man; and *Nina and the Neurons: Go Digital* on the CBeebies channel targeted at children aged 6 years and under, exploring topics 3D printing, coding and driverless cars.

5.4. Robotic kits

A growing number of robotic interfaces and platforms targeted to young children ages 8 years and younger are becoming available. While many of these are marketed as a STEM toy or tool, it is important to note that not all robotics kits actually involve a programming or computer science element.

Programmable robotics kits allow young children to explore the foundations of computer science in a hands-on way. Some robotic systems are programmed using tangible programming (Bers and Horn, 2010[157]; Horn, Crouser and Bers, 2011[158]) and others with block-based programming in screens. The use of educational robotics can be developmentally appropriate for early childhood education when it facilitates cognitive as well as fine motor and social development (Bers, 2007[159]; Clements, 1999[50]; Lee, Sullivan and Bers, 2013[78]; Wahlström et al., 2000[160]). Young children can become engineers by playing with motors and sensors as well as storytellers by creating and sharing personally meaningful projects that react in response to their environment (Bers, 2007[159]; Bers, 2018[9]). Thus, the use of robotic systems in early childhood has the potential to expand the range of computer science concepts and skills and include topics related to hardware and software, inputs, and outputs.

There are a growing number of commercially available introductory robotic systems for young children that introduce computer science and CT concepts (Table 2). For example, Code-a-Pillar, a robotic caterpillar toy created by the company Fisher Price, prompts preschool aged children to arrange (and rearrange) easy-to-connect segments (i.e. pieces of code) to decide where Code-a-Pillar should move. The Bee-Bot robot is also popular with preschool and early childhood students. The original Bee-Bot, designed to look like a friendly yellow bee, was programmed to move with the directional keys on its back. A newer version called Blue-Bot is transparent, allowing children to see and explore the technology inside the robot. Additionally, Blue-Bot is Bluetooth enabled and is compatible with tablets and computers. This allows children to plan algorithms on screen and send them remotely to the Blue-Bot to perform. A small study on Bee-Bot with 5 to 6-year-olds has found that interventions with the robot can lead to significant improvement in visual-

spatial working memory and inhibition skills (Di Lieto et al., 2017[161]). However, to programme it, children need to use screens as well as receive help from adults to manipulate the interface.

The KIBO robotics kit, developed by the DevTech Research Group at Tufts University and commercially available from KinderLab Robotics, offers young children ages 4 to 7 years an opportunity to explore building and engineering (through assembling their robot) as well as programming (using a tangible block language) without the need of screens or adult assistance. KIBO engages children with open-ended projects of their choice by reinforcing the design process while they build a mobile robot using wheels, motors, lights, and a variety of sensors. KIBO is programmed using interlocking wooden programming blocks. These wooden blocks contain no embedded electronics and are scanned by the KIBO robot. KIBO's design builds on extensive research on tangible programming that uses physical objects to represent the various aspects of computer programming (Horn and Bers, 2019[162]) KIBO's block programming language is composed of 21+ individual wooden programming blocks. Some of these blocks represent simple motions for the robot such as, move Forward, Backward, Spin, and Shake. Other blocks represent complex programming concepts such as Repeat Loops and Conditional "If" statements that involve sensor input. KIBO's design was based on years of research in collaboration with researchers, teachers, and early childhood experts to meet the learning needs of young children in a developmentally appropriate and fun way (Kazakoff and Bers, 2014[60]; Sullivan and Umashi Bers, 2016[163]; Sullivan, Elkin and Bers, 2015[66]). In addition to the tangible programming language, the KIBO robot comes with sensors and actuators (motors and light bulb and microphone/sound recorder), as well as art platforms. These modules can be interchangeably combined on the robot body. The use of sensors, such as light, distance and sound, is well aligned with early childhood curriculum that engages children in exploring both human and animal sensors. Motors can be connected to the sides or the top of the robot to enable mobility and rotation. All these elements increase the potential of children to create and imagine different projects that can move around and react to the environment (Elkin, Sullivan and Bers, 2018[164]; Sullivan, Bers and Mihm, 2017[65]; Sullivan, Elkin and Bers, 2015[66]).

Table 2.Commercially available robotic kits for young children that introduce computer science and CT concepts

Tool	*Interface*	*CT emphasis*	*Age range*
Bee-Bot	*Plastic "Bee" toy with movement buttons*	*Algorithms*	*4 years +*
Code-a-pillar	*Detachable plastic "caterpillar" body parts with coding instructions*	*Algorithms*	*3-6 years*
Code 'n Learn Kinderbot	*Plastic robot interface programmed with buttons on head. Free-play and challenge modes*	*Algorithms, Debugging*	*3-6 years*

Cubetto	*Moving Wooden robot with coded symbols on sequence board*	*Algorithms, Representation*	*3-6 years*
Dash and Dot	*Plastic robots coded with tablet app*	*Algorithms, Control Structures, Debugging*	*8-12 years*
KIBO Robot	*Plastic and wood robot coded with interlocking wooden barcode blocks*	*Design process, Algorithms, Representation, Control structures, Hardware/Software Debugging*	*3-8 years*
LEGO Coding Express	*Plastic train set with actions that can be coded with action bricks along the train's track*	*Design process, Algorithms, Control structures, Hardware/Software, Debugging*	*2 years +*
LEGO WeDo 2.0	*A kit of robotic and plastic building brick pieces coded with a tablet or computer app*	*Design process, Algorithms, Control structures, Hardware/Software, Debugging*	*7 years +*
Ozobot	*A robot that can be coded screen-free (using sensors that follow lines and read "colour codes" made with markers or sticker) or with an app*	*Algorithms, Control structures, Debugging*	*5 years+*

5.5. Unplugged activities and products

Organisations such as the World Health Organization (WHO) or the American Academy of Paediatrics (AAP) have issued recommendations related to young children's use of digital technology. The AAP calls for no screen time at all for children until 18 to 24 months, except for video chatting, and says children ages 2 to 5 should get an hour or less (Council on Communications and Media, 2016[165]). It has also developed the Family Media Use Plan for older children, in which parents and children negotiate limits and boundaries around screen usage. In its guidelines on physical activity, sedentary behaviour, and sleep for young children, WHO similarly recommends no screens for children under 2, and less than an hour a day for children 2 to 5 (World Health Organisation, 2019[166]). Aligned with these recommendations, this section presents unplugged and "low-tech" approaches for promoting CT.

5.5.1. Computer Science Unplugged

One of the guiding ideas behind the Computer Science Unplugged movement is that before engaging children in learning how to programme, it is important for them to learn basic CT

concepts, including how to decompose problems into smaller more manageable parts (decomposition), how to design precise steps to solve those problems (algorithms), and how to represent solutions into code – all of which can actually be explored without a computer (Bell and Vahrenhold, 2018[167]; Caeli and Yadav, 2019[168]).

Computer Science Unplugged has developed into a powerful global movement because it provides young children (and ECEC professionals) an approachable, hands-on, and screen-free way to explore computer science concepts. There is growing evidence that unplugged computer science activities are effective at teaching CT (Rodriguez et al., 2017[169]). Furthermore, unplugged approaches claim to enable the development of CT without spending time or cognitive resources on syntax and grammar of programming languages (Bell et al., 2009[170]; Bell and Vahrenhold, 2018[167]).

The original Computer Science Unplugged project was based at Canterbury University in New Zealand and has since been widely adopted internationally and recommended also in the ACM K-12 curriculum (Bell et al., 2009[170]). Computer Science Unplugged uses activities, games, magic tricks and more to introduce children to ways of thinking about computer science and to engage them in CT without reliance on learning computer programming. Unplugged activities place emphasis on promoting CT, rather than focusing on learning the syntax of a particular coding language. For example, an unplugged computer science activity in kindergarten might involve creating bead necklaces in binary numeric code with beads that represent 1s and 0s, using a grid and symbols to put classic fairy tales in a logical order or making a peanut butter sandwich following a set of instructions or algorithm.

Some activities that are described as "unplugged" are essentially coding exercises conducted offline using some of the same symbols and syntax as actual programming (Relkin and Strawhacker, 2021[171]). An example of this is how the website ScratchJr.org allows print out of large programming block cards that can be used to play a game called "Programmer Says". This game uses programming language instead of the verbal instructions to help students gain familiarity with coding commands (Relkin and Strawhacker, 2021[171]). Other resources teach CT-related principles without directly invoking coding commands. For example, CSunplugged.org's *Divide and Conquer?* uses animal playing cards to teach about algorithms and related concepts.

New research is constantly leading to revisions and refinements in educational practices around unplugged computer science. Some explores how unplugged coding activities (e.g. board game or paper-based coding) compare to unplugged CT activities (e.g. non-coding sorting and pattern matching) when employed in early childhood education (e.g. see (Barr and Stephenson, 2011[11]; Bell and Lodi, 2019[172]; Bell and Vahrenhold, 2018[167]; Upadhyaya, McGill and Decker, 2020[173]). Other examines the impact of CS Unplugged on young children. For example, while some studies have reported that unplugged activities do not increase interest or knowledge in CS/CT as much as traditional coding activities (Black et al., 2013[174]), others have found that unplugged lessons alone are just as effective (if not better) at promoting CT (Hermans and Aivaloglou, 2017[175]; Metin, 2020[176]; Wohl, Porter and Clinch, 2015[177]). Yet other studies have suggested that the most powerful way to promote CT in young children is to integrate unplugged exercises and coding activities together (Metin, 2020[176]; Huang and Looi, 2020[178]; Bers, 2020[36]; Thies and Vahrenhold, 2012[179]; Thies and Vahrenhold, 2013[180]).

5.5.2. Unplugged products and resources

The Computer Science Unplugged website offers a collection of free learning activities that teach computer science through engaging games and puzzles that use cards, string, crayons and lots of running around. This database of activities was developed with the intention

that young students could dive head first into computer science, experiencing the kinds of questions and challenges that computer scientists experience, but without having to learn programming first. None of the activities requires computers and focus on the use of arts, crafts, or physical activity. For example, the *Sorting Network* activity has teams of six running through a network drawn on the ground.

The activities published through the CS Unplugged website are widely used in classrooms and out-of-school instruction (Duncan and Bell, 2015[181])and have been translated into over 20 languages and used all around the world (Bell and Vahrenhold, 2018[167]). The unplugged approach is frequently mentioned in books on the teaching of computer science (Clarke, 2017[182]; Bers, 2021[3]) and used as a pedagogical technique on "coding" websites such as code.org. The CS Unplugged approach appears in curriculum recommendations, for instance as part of the design of a middle-years school curriculum (Schofield, Erlinger and Dodds, 2014[183])as a component of the *Exploring Computer Science* course (Goode and Margolis, 2011[184]),or as a resource to support the Australian Digital Technologies curriculum (Faulkner, 2015[185]).

Following the surge of the unplugged computer science movement, commercial companies began developing and marketing a new range of unplugged games and products. These offer a generally low-cost way to engage children with CT as compared with traditional technologies. For example, the Robot Turtles board game teaches coding concepts to children ages three and up and was the most backed board game in the history of the Kickstarter crowdfunding platform. Playing the game is easy: it involves creating a maze on the board with the turtles in the corners and the jewels in the centre. Young children then play instruction cards (such as, turn right, turn left, move forward, etc.) to "programme" their turtles to get to their jewels. The board can be set up differently each time and, as children get more familiar with the cards, more complex instructions can be used. This type of game engages young children in CT by having them create sequences and solve problems.

LittleCodr is a newer example of an unplugged product designed to foster CT to young children (ages 4-8) in an unplugged and "no-tech" capacity. LittleCodr, also originally funded by a Kickstarter campaign, is a card game that introduces the basics of programming by prompting young children to lay out a series of commands for other players (typically, an adult player) to act out.

6. Effective and scalable CT education

6.1. Overview of global CT initiatives

To date, most nationwide coding initiatives target children in primary and secondary levels of education, but a growing number of countries and regions have established clear policies and approaches for introducing technology and computer programming to young children (Australian Government Department of Education, Skills and Employment, 2015[186]; Department for Education, 2013[37]; Unahalekhaka and Govind, 2021[187]). This section, organised by global regions, outlines current CT initiatives in early education.

6.1.1. Americas

In the Americas, the United States is arguably leading the way for popularizing and implementing CT educational programmes, although many other nations are preparing to launch curricular or out-of-school initiatives. In April 2016, the White House launched a STEM initiative, including engineering and computer science, for early education (White

House, US, 2016[188]) by convening researchers, policy makers, industry, and educators. The US-based Code.org initiative developed by a non-profit organisation has encountered large success. Code.org aims to encourage people, particularly school-aged students, to learn computer science and practice coding skills during campaigns such as Hour of Code, which provides free resources for schools to engage students starting in kindergarten in free, 1-hour, ready-to-run curricular games, activities, and events.

In Canada, there is no specific mention of CT in provincial and territorial early learning frameworks. However, there are references in some curricular frameworks to related terms and practices, such as "technological competence" in kindergarten, referring to an understanding of technological applications and ability to apply appropriate technologies for solving (Gouvernment of Newfoundland and Labrador, 2015[189]). Policy frameworks such as the "Digital Action Plan for Education and Higher Education" (Ministère de l'Éducation et de l'Enseignement, 2018[190]) and the "Educating for a Digital World" report (Conseil supérieur de l'éducation, 2020[191]) outline plans for preparing Canadian schools, students, and teachers to emphasise coding and robotics in education starting from a young age. In the Canadian territory of British Columbia (BC), children ages 5-8 are supported by the BC Early Learning Framework and the BC curriculum. The BC curriculum includes Applied Design, Skills and Technologies, which supports CT for children. Further, many of the generalised CT skills of metacognition, creativity, critical and reflective thinking are embedded as core competences within the BC curriculum. Finally, Alberta, BC, and the Northwest Territories have outlined digital literacy frameworks beginning in kindergarten, which align philosophically with the principles of technology as a platform to support innovation and discovery for students, rather than purely an instructional aid for teachers (Gouvernment of Alberta, 2013[192]).

Other countries, including Chile, Argentina, Uruguay, and Brazil are all implementing national curriculum changes to include computation or digital technologies proficiencies in some way, with many specifically emphasising CT skills such as decomposition, pattern recognition, and abstraction starting in early childhood (Brackmann et al., 2016[193]).

6.1.2. Europe

Europe also has a wide array of CT initiatives underway. A survey of 21 participating European nations (Balanskat and Englehardt, 2014[194]) (reported that coding is already part of the curriculum at a national, regional, or local level in 16 countries: Austria, Bulgaria, Czech Republic, Denmark, Estonia, France, Hungary, Ireland, Israel, Lithuania, Malta, Spain, Poland, Portugal, Slovakia and the United Kingdom (England). The United Kingdom released a national curriculum framework in 2013 that included computing as an educational domain that needed to be addressed in school beginning in early childhood. In Finland, since 2016 all primary school students are required to learn programming (Pretz, 2014[195]). Many schools in Estonia are teaching programming to children as young starting at age 6, and countries like Italy are working on changing their curricula to include computer science and digital technologies (Jones, 2016[196]) (Pretz, 2014[195]) (Trevallion, 2014[197]).

In Spain, CT has been considered of great importance for several years, and it has been included in many educational schools and in the curricula of several regions. There are several ongoing national initiatives, for example the School of Computational Thinking (EPCIA), which includes activities for learners aged zero to five years (Spanish Government Ministry of Education and Vocational Training, 2021[198]). And at the regional level, Navarra has included CT content in primary education since 2018, integrating it into mathematics, while the regions of Madrid and Catalonia have created robotics and programming subjects in both primary and secondary education (Spanish Government,

Ministry of Education and Vocational Training, 2018[199]). Royal Decrees about the national curriculum include the development of digital competence and CT throughout all compulsory education, starting from ECEC. The digital competencies include, among others, the development of CT that begins in ECEC, finding solutions to simple technological problems (block programming, educational robotics) in primary education.

In Finland, CT is included in curriculum frameworks in ECEC, before children start primary education. Thinking and learning skills and multi-literacy and competence in information and communication technology are part of transversal competence in Finnish core curricula. The National Agency of Education's "Right to Learn: New literacy skills" development programme is designed to strengthen children's and young people's information and communication technology competence, media literacy and programming skills in early childhood education and in pre-primary and primary education (Kulju, Kupiainen and Pienimaki, 2020[200]). Additionally, the Ministry of Education and Culture has funded programmes related to new literacy skills in 22 municipalities, and in 2021 The Finnish National Agency for Education funded in-service training programmes and programmes which concentrate to develop innovative digital learning in over 25 municipalities.

In the Flemish Community of Belgium (Flanders), CT has been included as a separate building block within the key competence 'Digital competence and media literacy' in ongoing curriculum reforms in secondary education. Primary education attainment targets will be reconsidered at a later stage. In Germany, Bavaria is the only region to include specific guidance on how to develop children's media skills in its ECEC curriculum framework. However, the German initiative *Haus der kleinen Forscher* is very active in this area and informs pedagogues on how to promote ICT skills in ECEC centres (Gunther, 2017[201]). And in Italy, CT in ECEC and primary education is a topic of interest since the approval of a motion in the Chamber of Deputies in 2019 (Motion 1-00117 of Feb-Mar 2019).

6.1.3. Asia, Australia, and Pacific Island nations

In the Asia-Pacific region, countries such as Korea, Taiwan, Hong Kong, and China have all launched national curricular reforms to address the current movement in CT education (So, Jong and Liu, 2019[202]). Singapore launched nationwide projects to bring programming through a PlayMaker initiative that brings multiple technologies into early childhood classrooms (Digital News Asia, 2015[203]; Sullivan and Bers, 2017[99]).

Australia and New Zealand are working on changing their curricula to include computer science and digital technologies (Jones, 2016[196]; Australian Government Department of Education, Skills and Employment, 2015[186]; Pretz, 2014[195]; Trevallion, 2014[197]). In Australia, childcare services are required to base their educational programme on an 'approved learning framework'. This assists educators to address the developmental needs, interests, and experiences of each child, while taking into account individual differences. One of the framework's outcomes is for educators to support children to engage with information and communication technologies to access information, investigate ideas, and represent their thinking. Examples included in the framework are the use of technologies as a tool for designing and drawing, accessing images and information, and exploring diverse perspectives. As the two current frameworks have now been in use for close to a decade, an update has been commissioned to ensure they continue to reflect contemporary developments in practice and knowledge.

6.2. Professional development and qualifications of teachers and administrators

Research has shown that educators who work with young children have gaps in their knowledge about how to teach with technology to young children and how to successfully integrate technology in their practice (Bers, Seddighin and Sullivan, 2013[89]; Redmond et al., 2021[204]). Gaining the relevant technological and pedagogical skills is therefore emerging as a new demand placed on educators them (Yadav et al., 2016[205]; Bower and Falkner, 2015[206]) spoke of an "urgent and pressing need to [help] … teachers develop computational thinking pedagogies" and, referring to the introduction of the computing syllabus in the United Kingdom, (Dredge, 2014[207]) contended that, as with any major curricular change, "tens of thousands of primary schoolteachers who may be new to programming themselves" would be "at the sharp end" of implementation.

Demands relating to teaching digital technologies translate into challenges to initial teacher education. There is a growing need to provide pre-service teachers with the knowledge and dispositions to successfully incorporate CT into their curricula and practice in meaningful ways (Yadav et al., 2018[208]), and researchers have argued that teacher education needs to "provide not only the fundamentals of digital literacy… but also the computational thinking processes needed to understand the scientific practices that underpin technology" (Bower and Falkner, 2015[206]). Moreover, social inequities in children's access to technology required for remote instruction has added strain on the educators who serve them (Dubois, Bright and Laforce, 2021[209]). Beyond the logistical challenges, distance learning itself poses challenges in early childhood, when hands-on and play-based learning are essential (NAEYC, 2020[210]).

Pedagogical approaches for integrating technology and STEM in early childhood are emerging as a promising area of focus, but these efforts are still in early adoption stages (McClure, 2017[211]). Researchers have developed various theoretical frameworks to model technology integration in education. The Technology, Pedagogy, and Content Knowledge (TPACK) framework outlines the intersection of content knowledge and pedagogical approaches that can support educators to integrate technology effectively in their classrooms (Koehler and Mishra, 2009[212]). The Substitution Augmentation Modification Redefinition (SAMR) framework focuses more on the context in which technology differentially supports learning scenarios (Puentedura, 2013[213]). More recently, (Kimmons, Graham and West, 2020[214]) attempted to integrate learning context and instructional application through their theoretical PICRAT model, which examines students' engagement with a given technology (passive, interactive, or creative) and the way the technology integration influences the educator's practice before integration (replacement, amplification, transformation).

Regardless of theoretical framework or approach taken, early childhood educators need professional development to be successful in CT initiatives. A survey of pre-service elementary school teachers in the United States demonstrated that only 10 percent understood the concept of CT (Campbell, 2019[215]). Another study found that 75 percent of teachers incorrectly considered "creating documents or presentations on the computer" as a topic one would learn in a computer science course (Google/Gallup, 2015[41]). To address this gap in computer science and CT knowledge, educators need effective training and support. Research has shown that professional development training workshops can be an effective way to increase educators' confidence and competence in teaching these areas (e.g. (Bers, Seddighin and Sullivan, 2013[89]). Yadav et al. (2018[208]) found that professional development can help teachers better understand CT and how CT could be helpful in their classroom.

One example of a CT-focused professional development is the CT4EDU initiative in the United States. CT4EDU is a project funded by the National Science Foundation bringing together Michigan State University, Oakland Schools (Michigan) and the American Institute for Research to design, implement, and assess a high-quality, integrated curriculum, and professional development that supports elementary school teachers in embedding CT into their classrooms. In addition to developing several CT teaching resources, such as classroom posters, lesson screeners (to determine where CT concepts may already factor in existing math and science lessons), and a "toolkit" of core foundational concepts, definitions, and examples (e.g. debugging, abstraction), the initiative has also generated empirical research on effective CT professional development. Studies have identified elementary teacher perspectives and profiles for integrating CT into mathematics and science, and productive points of overlap in mathematics instruction and CT concepts (Rich, Yadav and Larimore, 2020[216]; Rich, Yadav and Schwarz, 2019[131]).

A growing number of post-graduate and certificate programmes, in-person workshops, and online professional development training aim at providing teachers with these skills in several OECD countries, although they tend to target more primary and secondary education teachers than those in pre-primary. Many of these programmes, and especially the longer ones, tend to be self-paced and offered in an online format. For example, the Teacher Engineering Education Programme (TEEP) at Tufts University is to be completed fully online in approximately 18 months. Similarly, the International Society for Technology Education (ISTE) Professional Certification programme is proposed online over 14 weeks and requires around 40 hours of work to build a portfolio. Another common feature of training programmes in this area is to provide opportunities for hands-on learning in combination with online content, as well as opportunities to test or showcase new knowledge.

There are also a growing number of short-term training programmes and workshops both online and in person. For example, Code.org offers in-person workshops that vary in duration based on topic and location, but that are typically completed in about eight hours. The Child Care Education Institute offers online professional development workshops that take approximately two self-paced hours to complete on different topics related to CT such as coding in early childhood education, robotics in early childhood education, and more.

6.3. Assessment and documentation

The 2018 International Computer and Information Literacy Study (ICILS) of the International Association for the Evaluation of Educational Achievement (IEA) investigated how well students in their eighth year of schooling are prepared for study, work, and life in a digital world. As part of this study, participating countries could administer an optional component to assess CT, which was defined as the "ability to recognise aspects of real-world problems which are appropriate for computational formulation and to evaluate and develop algorithmic solutions to those problems so that the solutions could be operationalised with a computer" (Fraillon et al., 2020[23]). The assessment of CT evaluated not only students' ability to analyse and break down a problem into logical steps but also their understanding of how computers might be used to solve a problem. Results from ICILS 2018 indicated that access to computers at home and experience using computers as well as socio-economic status were positively associated with student's CT skills. Additionally, students from non-immigrant families had significantly higher CT scale scores than students from non-immigrant families (Fraillon et al., 2020[23]).

Today, multiple tools exist to evaluate the effectiveness of initiatives and interventions for children to learn computer science and CT, and to gauge where children are in their

developmental progression. However, existing assessments are typically based on programming languages geared towards older children or teenagers and tend to provide formative feedback rather than being standardised assessments enabling comparisons across different contexts and age groups (de Ruiter and Bers, 2021[217]). For example, Quizly (Maiorana, Giordano and Morelli, 2015[218]) is based on the programming language AppInventor (Magnuson, 2010[219]) and geared towards middle and high school students. Quizly has been developed as a tool to help teachers to design problems for students and automatically compare student answers with model answers. As such, questions must be developed by the teachers, and are not standardised. Another tool, Dr. Scratch (Moreno-León and Robles, 2015[220]) automatically evaluates projects created in the programming language Scratch (Resnick et al., 2009[142]). However, Scratch is aimed for children eight years of age and above. Dr. Scratch can only evaluate what is provided by the user. This is different from an assessment that explicitly and purposefully tests different aspects of coding in the same way that, for example, reading assessments probe different aspects of reading ability (e.g. phonemic awareness, fluency, vocabulary) in an age-appropriate way.

Recently, efforts have been made to develop assessment instruments to evaluate CT skills more broadly, and with a specific focus on young children. For instance, the Coding Stages Assessment (CSA) is a new instrument that allows assessing young children's coding ability in the visual programming language ScratchJr (de Ruiter and Bers, 2021[217]). The CSA is an interactive, developmentally appropriate assessment for children from kindergarten (age five) through third grade (age eight). In line with the Coding as Another Language approach, the CSA assigns children to one of the five coding stages as laid out in the Coding Stages framework (Bers, 2019[119]), which draws parallels with literacy development: Emergent, Coding and Decoding, Fluency, New Knowledge, or Purposefulness (de Ruiter and Bers, 2021[217]).

Further, to understand children's developing coding skills in the context of their creative work, researchers have developed rubrics for project-based assessment of children's coding artefacts using the ScratchJr language and the KIBO robotic kit (Unahalekhaka and Bers, 2022[221]). Both rubrics evaluate programming concepts and design aspects to capture children's ability to transform their creative ideas into animated coding projects. Some of the rubrics' subcategories for programming concepts including repeat, events, coordination, and number parameters align with CT concepts such as flow control, logical thinking, synchronisation, and data representation (Unahalekhaka and Bers, 2022[222]). Studies looking at the evaluation of children's projects using the ScratchJr Project Rubric and the KIBO Project Rubric show that both assessment tools have good reliability and validity properties (Unahalekhaka and Bers, 2022[222]) (Govind and Bers, 2021[223]) (Callanan, Cervantes and Loomis, 2011[224]).

TechCheck (Relkin, de Ruiter and Bers, 2020[225]) (Relkin, de Ruiter and Bers, 2020[225]) (Relkin, 2021[226]) is an unplugged assessment of CT for children ages 5-9 designed to be used both cross-sectionally and longitudinally. *TechCheck* was validated through expert review by 19 computer science and child development experts (81% agreement among raters) and tested with a cohort of 768 students in the first years of primary education. A study had children engage with puzzle-like challenges that leveraged CT concepts but did not require coding experience to complete, and *TechCheck* scores correlated moderately with a previously validated, interview-based CT assessment tool. *TechCheck* probes six domains of CT: algorithms, modularity, control structures, representation, hardware/software and debugging. The assessment can be administered to individuals, whole classrooms or groups, either online or in person, in under 20 minutes, and rater feedback indicates ease of administration, ability to engage of children and simplicity of scoring. A new version (*TechCheck-2*) exists, and a validation study demonstrated its

good psychometric properties, including the ability to distinguish among young children with different CT abilities (Relkin, 2021[226]).

6.4. Informal learning spaces

Young children's CT can also be developed in informal learning spaces (Callanan, Cervantes and Loomis, 2011[224]) summarise five key dimensions of informal learning as being: 1) non-didactic, 2) highly socially collaborative, 3) embedded in meaningful activity, 4) initiated by learner's interest or choice, and 5) removed from external assessment. Accordingly, the informal learning spaces discussed here refer to environments that invite multiple pathways for attaining and transmitting knowledge, promote social and collaborative interactions, and engage children in meaningful and self-driven activities for the sake of enrichment, not evaluation. Examples of such spaces include children's homes, museums, libraries, community centres, after-school enrichment programmes, and other spaces that are accessible to young children and their caregivers.

In informal education settings, such as museums, homes, libraries, and out-of-school learning sites, CT learning may be most directly visible in activities described in research as "making" or "tinkering", learning experiences which leverage distributed knowledge, collaborative design processes, and constructionist "learning-by-making" pedagogies (Honey and Kanter, 2013[227]; Martin, 2015[228]; Papert and Harel, 1991[229]). Peer-supported making and tinkering activities have been shown to have a positive effect on youth because of the potential for "feedback-in-practice," which contributes to deep and transformative learning (DiGiacomo and Gutiérrez, 2015[230]).

The maker educational environment, or makerspace, is characterised by a blend of project-based pedagogical practices alongside informal "ways of seeing, valuing, thinking, and doing found in participatory cultures," which contributes to participant reports of makerspaces "feeling like a family or a group of friends" (Sheridan et al., 2014[231]). All of these cultural elements contribute to young makers who develop cognition, character, and social skills, as well as technical and professional attitudes (Agency by Design, 2015[232]). By intentionally designing an environment rich with technologies, tools, resources, and community values, makerspaces can provide makers with opportunities to develop identities as individuals and community members.

In the early childhood context, the Reggio Emilia approach has long focused on these issues. Loris Malaguzzi, founder of the Reggio Emilia pedagogy, coined the concept of the environment as a "third teacher" to capture the profound role that he believed environment plays in children's development, along with the "first" teachers, the child's caregivers, and the "second" teachers, the classroom educators (Biermeier, 2015[233]). In education communities, makerspaces have become sites to take up explorations of personally motivated problem solving and have been tied to 21st century learning outcomes of perseverance, creativity, persistence, and because of the emphasis on creation with digital tools, of CT (Iwata et al., 2020[234])

Bers (2021[3]) argues that the purpose of CT is to cultivate fluency with technological tools as a medium of expression, not necessarily as an end in itself. Computational making is part of this expression. A maker space provides the tools, community, and dedicated space for computational making to happen. A space combining expressive goals and a community for making things is inherently a space in which CT turns into computational making.

6.5. Family engagement

Research has long confirmed the critical role that parents and caregivers play as children's first teachers of play, learning, and healthy development (NAEYC and Fred Rogers Center, 2012[235]; Rideout, 2014[236]). As caregivers of today are increasingly tasked with exposing their young children to 21st century skills to prepare them for a global digital landscape, CT is becoming a household term and caregivers are (Bers, New and Boudreau, 2004[237]) prioritising CT as an early learning goal. In many ways, CT is still largely situated within the computer science discipline. However, as technology continues to grow and young children are increasingly exposed to a wide range of technological tools, CT is being treated more like the "universally applicable attitude and skill set" that Wing (2006[1]) and others purported it to be. Thus, CT skills that can be learned and fostered through young children's everyday play and learning activities, many of which occur in informal spaces in the presence of family members.

In the United States, the Family Coding Days project led by the DevTech Research Group first originated in the early 2000s as Project Inter-Actions, an exploration of intergenerational learning with robotics. Children between the ages of 4-7 and their parents attended a series of five-week workshops, during which they were introduced to programming using LEGO bricks. The project revealed several interesting findings about the ways in which children and parents learn about technology and engage with powerful ideas such as sequencing, looping, and debugging (Beals and Bers, 2006[238]; Bers, 2007[159]; Bers, New and Boudreau, 2004[237]). In particular, the project revealed how these workshops could generate a multigenerational "community of practice" (Lee et al., 2011[19]) that encourages families to engage with each other and with new knowledge and skills by producing creative computational artefacts.

This project was extended when it was piloted at in local schools and museums in the Boston area (Govind, 2019[239]). Children between five and seven years old as well as any family members ranging from grandparents to siblings were invited to attend these family-oriented programming events involving ScratchJr or KIBO. Using feedback from families' experiences, a detailed protocol was devised for hosting a family coding event with these tools and made freely accessible to anyone interested in facilitating this type of activity in their respective community. Between 2017 and 2018, 109 participants attended 14 ScratchJr or KIBO Family Coding Day events. The goals of these sessions were to help families learn about the technology, create a collaborative coding project, and share the project with peers. Findings from parent surveys and observations of play sessions indicated that these family coding events significantly enhanced both children and parents' interest in coding (Govind, 2019[239]). Regardless of whether parents worked in a STEM-related profession or what type of coding technology that families used, parents were able to successfully co-engage in coding projects by asking questions, offering suggestions and providing encouragement (Govind and Bers, 2020[240]; Relkin and Bers, 2020[25]; Relkin et al., 2020[241]).

Initiatives that engage young children and families in collaborative activities are well aligned with the principles of connected learning. Ito and colleagues define connected learning as "broadened access to learning that is socially embedded, interest-driven, and oriented towards educational, economic, or political opportunity" (Ito et al., 2013[242]). Collaborative computing activities, such as the ones described in the Family Coding Days project, capitalise on the interests of participants and are centred on production, inviting parents and children to co-design robotic creations or digital stories that are personally meaningful and interesting to them. These activities tend also to rely also peer-support and have a shared purpose, welcoming various opportunities for collaboration, feedback, and

community building. Finally, these activities can also be designed to be academically oriented and openly networked, offering children the opportunity to learn new skills and connect their learning across different settings.

6.6. Summary and recommendations

Taken together, the research summarised above can lead to practical advice and considerations for those hoping to implement CT initiatives in formal or school-based settings as well as in informal education settings. The following recommendations may be helpful for administrators, instructional leaders, and educators seeking to implement CT initiatives in formal early education settings:

- ***Offer CT tools that support children as creators with technology rather than as consumers of technology***. The literature highlights the many ways that children of today's increasingly global and technology-rich society are interacting with the technological tools around them. However, not all technology is the same. Some tools are made for consuming (e.g. televisions; passive digital games); others are made for creating (e.g. open-ended programming environments). Choosing open-ended and creative tools such as programmable robotics kits and open-ended programming languages can help to effectively support CT skills in young children.
- ***Invest in developmentally appropriate tools***. It is important for early childhood initiatives and curricula to choose tools specifically designed for young children. When choosing to dedicate resources to purchase and deploy new tools and technologies, educators and administrators may want to ask: Can young children effectively engage with these tools from a fine motor and cognitive perspective? Do they require reading ability or significant adult scaffolding to be used with young children? How can the usage of these tools evolve as children grow older and progress in ECEC and school? How can these tools be used over the years as technologies continue to evolve?
- ***Provide adequate and ongoing training and support for ECEC professionals***. Research has shown that staff require training and support in order to effectively promote CT and computer science education in the early years. Before and during any new CT initiatives, professional development workshops specific to the tools and pedagogies teachers and staff are expected to implement should be provided. Considering self-paced and ongoing programmes, mentorships with "expert" staff and periodic check-ins can be especially useful.
- ***Provide time for planning and implementation***. ECEC staff in different roles will also need adequate time for planning their new curriculum and finding ways to meaningfully inject CT into their existing curriculum and activities. Teachers should be encouraged to think of ways to integrate CT across curricular domains (e.g. a STEAM approach to education) and to collaborate with educators specialising in other domains instead of trying to find additional time in an already tight schedule.

In informal learning spaces, the following practical considerations might be helpful for families and facilitators seeking to promote young children's coding and CT engagement:

Prioritise CT tools encouraging children to create with rather than to merely consume technology. As mentioned in the recommendations for formal education settings, it is equally important to consider offering open-ended and creative tools in informal education settings. The ways in which families can foster children's CT through those open-ended and creative technologies will be more engaged than with the passive "consumer"

technologies. In any space that is "family-friendly", parents and caregivers might consider the following questions: What might my child do or say while they are navigating this space and using these tools? What might I be doing or saying in turn? How does the technology and the technology-mediated activity enable us to co-engage in CT? Facilitators and designers of family-friendly environments might examine the variety of technological tools in their spaces and think carefully about the kinds of interactions those tools might provide.

Facilitating bidirectional home-school connections. Families can drive their own learning process about understanding what CT is and how it can be fostered through both unplugged and technology-mediated activities. As computer science education becomes an increasingly important national and international priority in schools and other formal learning settings, continuing children's coding and CT learning in informal settings through family engagement initiatives will be increasingly salient. Stakeholders who play a role in facilitating children's informal and formal learning experiences might consider the following questions: What activities might parents already be doing in homes and informal learning spaces that foster children's CT, and how can we empower parents to recognise and extend those activities? What technological tools might be introduced in school settings and how is that learning being shared with families?

Leveraging community resources. Facilitators should be encouraged to leverage the existing resources within communities when planning workshops, activities, and other events intended to support families' coding and CT engagement. In addition to the kinds of tools and technologies, it is crucial to think about how the community will utilise the space and the resources needed to make the opportunity accessible and engaging for all attendees. Facilitators might consider the following questions: Are there enough tools or materials for all families? What resources may be needed to enhance accessibility and inclusion? How are we supporting children and families from diverse backgrounds (e.g. with a different home language, with special needs) with this tool and activity?

7. Equity and access

7.1. Increasing diversity, access, equity, and inclusion in the fields of computational thinking and computer science

With the rise of global technological innovations has come a rapidly growing gender and racial divide within technology and engineering related fields, as well as within technical STEM fields more broadly.

Representation in technical STEM fields continue to lack diversity, despite programmes and interventions that have been initiated across OECD countries and others focusing on reaching a wider range of students, most often in secondary or early years of tertiary education. However, it has become clear that computer programming needs to be taught at earlier ages to more effectively prevent and address negative STEM stereotypes (Markert, 1996[243]; Sullivan, 2019[35]). Increasing interest in the information technology professions is therefore seen an important objective of the inclusion of computer science education the early years and the primary and secondary levels curricula (National Research Council, 2011[244]).

While a deep dive into these issues is beyond the scope of this review, the following sections highlight issues of diversity, equity, and inclusion in CT as it specifically relates to early childhood education and interventions.

7.2. Socio-economic inequalities in access to CT tools

When it emerged in recent decades, the term "digital divide" referred mainly to the socio-economic divide between those who had access to Internet-enabled digital devices and those who did not. Today, Internet access is generally widespread in educational settings in OECD countries, many of which are close to meeting the Sustainable Development Goal targets of ensuring that all schools have access to the Internet for pedagogical purposes, and of mobile network coverage (Burns and Gottschalk, 2019[245]).

Today, however, Internet-enabled devices and smart phones barely scratch the surface of the types of technologies available to support education, beginning in early childhood. As noted earlier in this review, there are now many digital tools, such as the robotics kits and tablet applications, available for young children to explore CT and computer science, and research has shown the many cognitive and social benefits that early exposure to such tools can have when used with a pedagogically sound approach. But many of these new tools, despite their benefits, are inaccessible due to the cost, the technical support, and the professional development needed for adequate implementation. Even free coding applications and games require schools or homes to have one-to-one (1:1) access to tablets or computers to be used as intended. The costs of these devices alone are already prohibitive to many, and that is without factoring in fees and time for training and professional development for educators to feel confident using these tools with young children.

The stark costs of new coding and engineering materials for young children has opened the door to a new type of digital divide. Now that most homes and schools do have Internet connectivity basic hardware, this phrase has taken on a new meaning. There is now a socio-economic gap between those with access to high-quality, open-ended, software and technology that promotes creative STEAM learning and those that do not (Sullivan and Strawhacker, 2021[94]). For example, access to computer science classes and clubs is generally lowest for students from lower-income households (Google/Gallup, 2015[41]). Unequal access to computer science education could place these students at a disadvantage as computer technology continues to advance, especially as coding is thought of as today's "new literacy" today (Bers, 2018[9]).

This new version of the digital divide is continually highlighted in empirical research. In a recent analysis of data from ICILS 2018, which tested over 46,000 students from 14 countries, researchers found that persistent gaps among students' CT performance were linked to their family's socio-economic backgrounds (Fraillon et al., 2020[23]; Karpiński, Di Pietro and Biagi, 2021[246]). Specifically, results "consistently showed that students from less advantaged backgrounds had lower levels of computer skills than those from more advantaged backgrounds, especially in CT" (Karpiński, Di Pietro and Biagi, 2021, p. 1[246]).

To address this divide, it is critical that adequate national level funding and support are provided to ECEC settings and schools. It may also be important to focus on unplugged CT curriculum in areas where the cost of other technologies is not feasible. Preliminary research has shown that unplugged activities may be useful for addressing these gaps by laying a foundation for later technology-mediated computer science learning (Bers, 2020[36]) (del Olmo-Muñoz, Cózar-Gutiérrez and González-Calero, 2020[247]). Young children aged 4-8 years, with their developmental need for physical, hands-on play and limited screen engagement, may benefit the most from foundational unplugged CT experiences (Przybylski and Weinstein, 2017[248]; Saxena, Baber and Kumar, 2020[249]).

7.3. Addressing issues with underrepresented groups in CT

Limited gender and ethnic diversity in technological industries that rely on CT and computer science skills continues to be a problem across OECD countries. Recent analyses of data from the Higher Education Statistics Authority (HESA) in the United Kingdom reveal "unacceptable" ethnic disparities in higher education STEM fields over the past 10 years, as well as in the pool of researchers eligible for the Royal Society's early career fellowship grants (The Royal Society, 2021[42]). Similar issues with racial diversity are evident in other OECD countries. In the United States, Caucasian men constitute approximately half of the scientists and engineers employed in science and engineering occupations, with both Asian men and women being also highly represented in the STEM workforce (National Center for Science and Engineering Statistics, 2017[250]).

Gender diversity is also a consistent issue in fields that rely on CT and computer science in many OECD countries. Women account for less than 20% of entrants into tertiary level computer science programmes across OECD countries and only around 18% of engineering entrants (OECD, 2017[251]). It has been long theorised that stereotype threat may influence the participation of women and ethnic minorities in STEM fields. Stereotype threat refers to the anxiety that one's performance on a task or activity will be seen through the lens of a negative stereotype (Steele, 1997[252]). For example, Spencer, Steele and Quinn (1999[253]) found that women performed significantly worse on a math test if they were first shown information indicating that women do not perform as highly as men on math tasks (to induce the negative stereotype). If the negative stereotype was not triggered (i.e. participants were told that there were no gender differences associated with the math test) women and men performed similarly on the test.

While most research on the influence of stereotype threat has focused on adolescent and adult research participants, research and developmental literature has shown that basic stereotypes do begin to develop in children around two to three years of age (Kuhn, Nash and Brucken, 1978[254]; Signorella, Bigler and Liben, 1993[255]). As children grow older, stereotypes about sports, occupations and adult roles expand, and their gender associations become more sophisticated (Sinno and Killen, 2009[256]). It is important for adults to be aware of these newly forming stereotypes to expand on them (or disprove them) by providing children with different role models, experiences and media that can help shift children's belief system (Sullivan, 2021[257]).

Early experiences have the potential to play an ongoing role in children's sense of belonging and confidence in different computer science or STEM activities and in their own developing identity as they grow up. Forming a positive "STEM identity" (Aschbacher, Li and Roth, 2009[258]) during this time can be pivotal to maintaining girls' interest in these fields. Prior research has shown that early childhood experiences with technology and engineering – or lack thereof – can continue to impact adolescents during middle school and high school, even those on competitive robotics and programming teams (Sullivan and Bers, 2019[44]). Children who are exposed to STEM curriculum and programming at an early age demonstrate fewer gender-based stereotypes regarding STEM careers, an increased interest in engineering, and fewer obstacles entering these fields later in life (Markert, 1996[243]; McLaren, 2009[259]; Steele, 1997[252]; Sullivan and Bers, 2017[99]; Sullivan, 2019[35]). Taken together with the past body of work on stereotypes, it is critical to begin reaching female children and those from racial or ethnic groups that are underrepresented in STEM with positive, developmentally appropriate experiences with CT, and computer science in general, from an early age (Sullivan and Bers, 2019[44]; Sullivan and Strawhacker, 2021[94]).

7.4. Disabilities and accessibility

Over 1 billion people, or 15% of the world's population, have some kind of disability. One-fifth of the estimated global total, or between 110 million and 190 million people, experience significant disabilities (World Health Organization and World Bank, 2011[260]). This impacts students across OECD countries as well. For example, 14% of public-school students in the United States ages 3-21 receive special education services under the Individuals with Disabilities Education Act (Congress, 1975) for some form of disability, which can range from a specific learning disorder, to a speech impairment, to autism (National Center for Education Statistics, 2022[261]).

When it comes to digital technology and issues of equity, it is important to consider children with disabilities and the various accessibility issues with new computing interfaces. As ECEC systems implement initiatives that bring computer science to young children, they face heightened demands for supporting ECEC professionals in meeting the needs of diverse groups of children.

Researchers and educators are increasingly building an argument of the benefits of having teachers expose and engage children with disabilities in CT and computer science (Bouck and Yadav, 2020[262]). Unfortunately, there is still insufficient research on best practices regarding access and exposure to CT and computer science for children with different types of cognitive, physical, emotional, and behavioural challenges. Most research on CT instruction for students with special needs has focused on students with low-incidence disabilities and autism, and much of this research focuses on educational pedagogies based around explicit instruction (Taylor, 2018[263]). These evidence-based explicit instruction pedagogies used by special educators contrast with the constructionist pedagogies advocated by researchers in the field of CT (Bers, 2020[36]; Levinson, Hunt and Hassenfeld, 2021[264]). While constructionist models allow for student-driven play to drive learning, explicit instruction provides a structure for learning. Using evidence-based explicit instruction, computer programming has been taught to students with Down syndrome, autism, and intellectual disability (Bouck and Yadav, 2020[262]; Knight, Wright and DeFreese, 2019[265]; Muñoz et al., 2018[266]; Pivetti et al., 2020[267]; Taylor, Vasquez and Donehower, 2017[268]).

Additionally, a growing number of educators have shown support for teaching and learning of CT through a Universal design for learning (UDL) approach. UDL is an instructional planning framework for meaningfully engaging a range of learners, including children with special needs, by proactively addressing barriers to learning (Center for Applied Special Technology, 2011[269]; Rose and Meyer, 2002[270]). There is research demonstrating the educational efficacy of teaching through the UDL framework (e.g. (Marino et al., 2013[271]; Rappolt-Schlichtmann et al., 2013[272]). Within the context of computing education, UDL can serve as the instructional framework in which teachers can embed the necessary supports, technologies, and strategies that lead to effective instruction for a broad range of learners.

8. Concluding remarks

The early years of development are an exciting and yet challenging period of growth for researchers, educators, and policymakers to consider when designing interfaces, curricula, and frameworks to support CT. Tools must be carefully selected for children who cannot yet read and write, who have a short attention span and working memory, who are honest in expressing engagement and frustration, who are just learning how to work with others, and who are eager to explore the world by touching, making, and breaking (Bers, 2021[3]).

The theories and empirical research reviewed in this document highlight that early childhood is a critical time in development to build on children's natural curiosity and support their newly developing CT skills and abilities. Studies conducted worldwide have shown how diverse children can learn with and about computer science, how this new discipline can help them make connections to more traditional domains of learning, and how CT can support cognitive and social development in general (Bers, 2020[36]) (Bers, 2021[3]).

This report has also summarised the importance of early CT education from a diversity, equity, and inclusion perspective (Markert, 1996[243]; Sullivan, 2019[35]; Karpiński, Di Pietro and Biagi, 2021[246]). There is a growing need for more diverse voices, truly representative of the global community, to be heard in the STEM and computer science fields that constitute major drivers of innovation. Early (and continued) exposure of all children to CT from a young age is critical to making this possible (Sullivan, 2021[257]) (Sullivan, 2019[35]; Sullivan and Strawhacker, 2021[94]).

8.1. Key takeaways for policymakers

This review has summarised theoretical contributions, a survey of commercially available technologies, curriculum development, frameworks, and empirical research focused on CT and ECEC. While research in this area has flourished in recent years, there is still a need for more robust scientific studies –including extensive randomised trials– to generate more conclusive evidence on the effects of CT educational programmes and interventions, as well as on the conditions for their potential implementation at scale. Taken together, this body of international work points to the following key takeaways for policymakers and other stakeholders:

- The foundational early childhood years (ages 3-8) are a critical time in development when it comes to fostering CT and computer science education. Early exposure to CT is also important from a social equity perspective to prevent stereotypes and ensure all young children receive equal opportunities to develop their digital literacy.
- Young children can master a range of CT concepts and skills including algorithms, modularity, control structures, representation, hardware/software, the design process, and debugging.
- There is a growing demand for countries to incorporate some form of computer science or CT into their curricula and learning frameworks for early levels of education. These can be helpful for increasing access to quality CT education. However, the use of digital technology in ECEC should add to children's experiences rather than replace interactions with traditional learning materials and games.
- Choosing developmentally appropriate tools for young learners is important for the success of any CT initiative. Policymakers should consider play-based, screen-free technologies and other "unplugged" approaches when creating programmes for very young learners to align with research recommendations around early learning and development, and about limited screen time.
- ECEC staff and leaders require tailored professional development and support to be successful in integrating CT into their work. Policymakers should consider allotting resources for training staff as part of any new initiative with CT.

- Parents, caregivers, and families are children's first teachers of play, learning, and healthy development. As such, policy makers should consider family engagement and resources as part of any new initiative with CT.
- More scientific research is needed to guide policy and practice about CT education for young children, including on the relationship between CT skills and other early cognitive and socio-emotional outcomes, and on the factors that may support the large scale deployment of proven tools and approaches.

References

Agency by Design (2015), *Maker-centered learning and the development of self: Preliminary findings of the Agency by Design Project*, http://www.pz.harvard.edu/sites/default/files/Maker-Centered-Learning-and-the-Development-of-Self_AbD_Jan-2015.pdf (accessed on November 2021). [232]

Aguirre-Muñoz, Z. and M. Pantoya (2016), "Engineering Literacy and Engagement in Kindergarten Classrooms", *Journal of Engineering Education*, Vol. 105/4, pp. 630-654, https://doi.org/10.1002/jee.20151. [122]

Aho, A. (2011), "Ubiquity symposium: Computation and Computational Thinking", *Ubiquity*, Vol. 2011/January, https://doi.org/10.1145/1922681.1922682. [20]

Aldemir, J. and H. Kermani (2016), "Integrated STEM curriculum: improving educational outcomes for Head Start children", *Early Child Development and Care*, Vol. 187/11, pp. 1694-1706, https://doi.org/10.1080/03004430.2016.1185102. [87]

Allen-Handy, A. et al. (2020), "Black Girls STEAMing Through Dance", in *Challenges and Opportunities for Transforming From STEM to STEAM Education, Advances in Educational Technologies and Instructional Design*, IGI Global, https://doi.org/10.4018/978-1-7998-2517-3.ch008. [91]

Aschbacher, P., E. Li and E. Roth (2009), "Is science me? High school students' identities, participation and aspirations in science, engineering, and medicine", *Journal of Research in Science Teaching*, Vol. 47/5, pp. 564-582, https://doi.org/10.1002/tea.20353. [258]

Atmatzidou, S. and S. Demetriadis (2016), "Advancing students' computational thinking skills through educational robotics: A study on age and gender relevant differences", *Robotics and Autonomous Systems*, Vol. 75, pp. 661-670, https://doi.org/10.1016/j.robot.2015.10.008. [34]

Australian Government Department of Education, Skills and Employment (2015), *Taking action now to revitalise STEM study in schools*, Minister's Media Centre website, https://ministers.dese.gov.au/pyne/taking-action-now-revitalise-stem-study-schools. [186]

Bailey, D. and J. Borwein (2011), "High-precision numerical integration: Progress and challenges", *Journal of Symbolic Computation*, Vol. 46/7, pp. 741-754, https://doi.org/10.1016/j.jsc.2010.08.010. [100]

Balanskat, A. and K. Englehardt (2014), *Computing our Future: Computer programming and coding - Priorities, school curricula and initiatives across Europe*, http://d3780a64-1081-4488-8549-6033200e3c03 (eun.org). [194]

Barnes, J. et al. (2017), *The influence of robot design on acceptance of social robots*, IEEE, https://doi.org/10.1109/urai.2017.7992883. [95]

Barr, D., J. Harrison and L. Conery (2011), "Computational thinking: A digital skill for everyone", *Learning & Leading with Technology*, Vol. 38/6, pp. 20-23. [18]

Barr, V. and C. Stephenson (2011), "Bringing computational thinking to K-12", *ACM Inroads*, Vol. 2/1, pp. 48-54, https://doi.org/10.1145/1929887.1929905. [11]

Beals, L. and M. Bers (2006), "Robotic Technologies: When parents put their learning ahead of their child's", *Journal of Interactive Learning*, Vol. 17/4, pp. 3411-366. [238]

Bell, T. et al. (2009), "Computer Science Unplugged: school students doing real computing", *The New Zealand Journal of Applied Computing and Information Technology*, Vol. 13, pp. 20-29. [170]

Bell, T. and M. Lodi (2019), "Constructing computational thinking without using computers", *n Constructivist Foundations*, Vol. 14/3, pp. 342-351, https://constructivist.info/14/3/342.bell. [172]

Bell, T. and J. Vahrenhold (2018), "CS Unplugged—How Is It Used, and Does It Work?", in *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications, Lecture Notes in Computer Science*, Springer International Publishing, Cham, https://doi.org/10.1007/978-3-319-98355-4_29. [167]

Bers, M. (2022), *Beyond Coding. How Children Learn Human Values through Programming*, MIT Press. [72]

Bers, M. (ed.) (2021), *Teaching Computational Thinking and Coding to Young Children*, IGI Global, https://doi.org/10.4018/978-1-7998-7308-2. [3]

Bers, M. (2020), *Coding as a Playground*, Routledge, https://doi.org/10.4324/9781003022602. [36]

Bers, M. (2019), "Coding as another language: a pedagogical approach for teaching computer science in early childhood", *Journal of Computers in Education*, Vol. 6/4, pp. 499-528, https://doi.org/10.1007/s40692-019-00147-3. [124]

Bers, M. (2019), "Coding as another language: Why computer science in early childhood should not be STEM", in Donohue, C. (ed.), *Exploring Key Issues in Early Childhood and Technology: Evolving Perspectives and Innovative Approaches*, Routledge, New York, NY, https://doi.org/10.4324/9780429457425. [119]

Bers, M. (2018), "Coding and Computational Thinking in Early Childhood: The Impact of ScratchJr in Europe", *European Journal of STEM Education*, Vol. 3/3, https://doi.org/10.20897/ejsteme/3868. [9]

Bers, M. (2012), *Designing Digital Experiences for Positive Youth Development: From Playpen to Playground*, Oxford. [81]

Bers, M. (2010), "The TangibleK Robotics Program: Applied Computational Thinking for Young Children", *Early Childhood Research and Practice*, Vol. 12/2, http://ecrp.uiuc.edu/v12n2/bers.html. [22]

Bers, M. (2008), *Blocks to Robots: Learning with Technology in the Early Childhood Classroom*, Teachers College Press. [82]

Bers, M. (2007), "Project InterActions: A Multigenerational Robotic Learning Environment", *Journal of Science Education and Technology*, Vol. 16/6, pp. 537-552, https://doi.org/10.1007/s10956-007-9074-2. [159]

Bers, M., R. New and L. Boudreau (2004), "Teaching and Learning when no one is Expert: Children and Parents Explore Technology", *Early Childhood Research and Practice*, Vol. 6/2, https://files.eric.ed.gov/fulltext/EJ1084881.pdf. [237]

Bers, M. and M. Resnick (2015), *Official ScratchJr Book: Help Your Kids Learn to Code*, No Starch Press, https://nostarch.com/scratchjr. [154]

Bers, M., S. Seddighin and A. Sullivan (2013), "Ready for Robotics: Bringing Together the T and E of STEM in Early Childhood Teacher Education", *Journal of Technology and Teacher Education*, Vol. 21/3, pp. 355-377, https://sites.tufts.edu/devtech/files/2018/02/BringingTogetherT.pdf. [89]

Berson, R. and M. Berson (eds.) (2010), *Tangible programming in early childhood: Revisiting developmental assumptions through new technologies*, Information Age Publishing. [157]

Biermeier (2015), "Inspired by Reggio Emilia: Emergent Curriculum in Relationship-Driven Learning Environments", Vol. 70/5, pp. 72 - 75, https://www.naeyc.org/resources/pubs/yc/nov2015/emergent-curriculum. [233]

Black, J. et al. (2013), "Making computing interesting to school students", *Proceedings of the 18th ACM conference on Innovation and technology in computer science education - ITiCSE '13*, https://doi.org/10.1145/2462476.2466519. [174]

Blair, C. and A. Diamond (2008), "Biological processes in prevention and intervention: The promotion of self-regulation as a means of preventing school failure", *Development and Psychopathology*, Vol. 20/3, pp. 899-911, https://doi.org/10.1017/s0954579408000436. [68]

Blikstein, P. and U. Wilensky (2009), "An Atom is Known by the Company it Keeps: A Constructionist Learning Environment for Materials Science Using Agent-Based Modeling", *International Journal of Computers for Mathematical Learning*, Vol. 14/2, pp. 81-119, https://doi.org/10.1007/s10758-009-9148-8. [101]

Bouck, E. and A. Yadav (2020), "Providing Access and Opportunity for Computational Thinking and Computer Science to Support Mathematics for Students With Disabilities", *Journal of Special Education Technology*, Vol. 37/1, pp. 151-160, https://doi.org/10.1177/0162643420978564. [262]

Bower, M. and K. Falkner (2015), "Computational thinking, the notional machine, pre-service teachers, and research opportunities", in D'Souza, D. and K. Falkner (eds.), *Proceedings of the 17th Australasian Computing Education Conference (ACE 2015)*, Australian Computer Society, http://crpit.com/confpapers/CRPITV160Bower.pdf. [206]

Brackmann, C. et al. (2016), "Computational thinking: Panorama of the Americas", *2016 International Symposium on Computers in Education (SIIE)*, https://doi.org/10.1109/siie.2016.7751839. [193]

Bravo Sánchez, F., A. González Correal and E. González Guerrero (2017), "Interactive Drama With Robots for Teaching Non-Technical Subjects", *Journal of Human-Robot Interaction*, Vol. 6/2, p. 48, https://doi.org/10.5898/jhri.6.2.bravo. [96]

Bredekamp, S. (1992), "What is "Developmentally Appropriate" and Why is it Important?", *Journal of Physical Education, Recreation & Dance*, Vol. 63/6, pp. 31-32, https://doi.org/10.1080/07303084.1992.10606612. [32]

Brosterman, N. (1997), *Inventing kindergarten*, New York: H. N. Abrams. [63]

Bruner, J. (2009), *The Process of Education (Revised edition)*, Harvard University Press. [132]

Burns, T. and F. Gottschalk (eds.) (2019), *Educating 21st Century Children: Emotional Well-being in the Digital Age*, Educational Research and Innovation, OECD Publishing, Paris, https://dx.doi.org/10.1787/b7f33425-en. [245]

Caeli, E. and A. Yadav (2019), "Unplugged Approaches to Computational Thinking: a Historical Perspective", *TechTrends*, Vol. 64/1, pp. 29-36, https://doi.org/10.1007/s11528-019-00410-5. [168]

Callanan, M., C. Cervantes and M. Loomis (2011), "Informal learning", *Wiley Interdisciplinary Reviews: Cognitive Science*, Vol. 2/6, pp. 646-655, https://doi.org/10.1002/wcs.143. [224]

Campbell, L. (2019), "Building Computational Thinking", in *Recruiting, Preparing, and Retaining STEM Teachers for a Global Generation*, BRILL, https://doi.org/10.1163/9789004399990_007. [215]

Cellan-Jones, R. (2019), *Computing in schools in 'steep decline'*, https://www.bbc.com/news/technology-48188877 (accessed on 17 May 2022). [43]

Center for Applied Special Technology (2011), *Universal Design for Learning Guidelines version 2.2*. [269]

Center on the Developing Child at Harvard University (2011), "Building the Brain's "Air Traffic Control" System: How Early Experiences Shape the Development of Executive Function", No. 11, Harvard, MA, http://www.developingchild.harvard.edu. [67]

Chinn, C. and B. Malhotra (2002), "Epistemologically authentic inquiry in schools: A theoretical framework for evaluating inquiry tasks", *Science Education*, Vol. 86/2, pp. 175-218, https://doi.org/10.1002/sce.10001. [110]

Çiftci, S. and A. Bildiren (2019), "The effect of coding courses on the cognitive abilities and problem-solving skills of preschool children", *Computer Science Education*, Vol. 30/1, pp. 3-21, https://doi.org/10.1080/08993408.2019.1696169. [55]

Clarke, B. (2017), *Computer Science Teacher: Insight into the computing classroom*, BCS Learning Development Limited, Swindon. [182]

Clements, D. (1999), "Subitizing: What Is It? Why Teach It?", *Teaching Children Mathematics*, Vol. 5/7, pp. 400-405, https://doi.org/10.5951/tcm.5.7.0400. [50]

Code.org (2021), *Glossary. CS Fundamentals for Elementary Schools (Grade K-5)*, https://code.org/curriculum/docs/k-5/glossary. [10]

Code.org (2018), *Code.org 2018 Annual Report.*, https://code.org/about/2018. [4]

Confrey, J. (1994), "A theory of intellectual development", *For the Learning of Mathematics*, Vol. 14/3, pp. 2-8, https://www.jstor.org/stable/40248118. [111]

Conseil supérieur de l'éducation (2020), *Educating For A Digital World: Report on the State and Needs of Education 2018-2020*, Conseil supérieur de l'éducation, Quebec, https://www.cse.gouv.qc.ca/en/educating-for-a-digital-world/. [191]

Council on Communications and Media (2016), "Media use in school-aged children and adolescents", *Pediatrics*, Vol. 138/5, https://doi.org/10.1542/peds.2016-2592. [165]

Csikszentmihalyi, M. (1981), "Some paradoxes in the definition of play", in Cheska, A. (ed.), *Play as Context*, Leisure Press, NY. [149]

Cunha, F. and J. Heckman (2007), "The Technology of Skill Formation", *American Economic Review*, Vol. 97/2, pp. 31-47, https://doi.org/10.1257/aer.97.2.31. [26]

Davee, S. et al. (2015), *Youth Makerspace Playbook*, https://makered.org/wp-content/uploads/2015/09/Youth-Makerspace-Playbook_FINAL.pdf (accessed on 17 May 2022). [141]

de Ruiter, L. and M. Bers (2021), "The Coding Stages Assessment: development and validation of an instrument for assessing young children's proficiency in the ScratchJr programming language", *Computer Science Education*, pp. 1-30, https://doi.org/10.1080/08993408.2021.1956216. [217]

del Olmo-Muñoz, J., R. Cózar-Gutiérrez and J. González-Calero (2020), "Computational thinking through unplugged activities in early years of Primary Education", *Computers & Education*, Vol. 150, https://doi.org/10.1016/j.compedu.2020.103832. [247]

Department for Education (2013), *National curriculum in England: computing programmes of study*, https://www.gov.uk/government/publications/national-curriculum-in-england-computing-programmes-of-study/national-curriculum-in-england-computing-programmes-of-study (accessed on 17 May 2022). [37]

Di Lieto, M. et al. (2017), "Educational Robotics intervention on Executive Functions in preschool children: A pilot study", *Computers in Human Behavior*, Vol. 71, pp. 16-23, https://doi.org/10.1016/j.chb.2017.01.018. [161]

DiGiacomo, D. and K. Gutiérrez (2015), "Relational Equity as a Design Tool Within Making and Tinkering Activities", *Mind, Culture, and Activity*, Vol. 23/2, pp. 141-153, https://doi.org/10.1080/10749039.2015.1058398. [230]

Digital News Asia (2015), *IDA launches $1.5m pilot to roll out tech toys for preschoolers*, https://www.digitalnewsasia.com/digital-economy/ida-launches-pilot-to-roll-out-tech-toys-for-preschoolers. [203]

Dodig-Crnkovic, G. (2002), "Scientific methods in computer science", in *Proceedings of the Conference for the Promotion of Research in IT at New Universities and at University Colleges in Sweden*, Skövde. [8]

Dredge, S. (2014), "Coding at school: a parent's guide to England's new computing curriculum", *The Guardian*, https://www.theguardian.com/technology/2014/sep/04/coding-school-computing-children-programming. [207]

Dubois, E., D. Bright and S. Laforce (2021), "Educating Minoritized Students in the United States During COVID-19: How Technology Can be Both the Problem and the Solution", *IT Professional*, Vol. 23/2, pp. 12-18, https://doi.org/10.1109/mitp.2021.3062765. [209]

Duncan, C. and T. Bell (2015), "A Pilot Computer Science and Programming Course for Primary School Students", *Proceedings of the Workshop in Primary and Secondary Computing Education*, https://doi.org/10.1145/2818314.2818328. [181]

Elkin, M., A. Sullivan and M. Bers (2018), "Books, butterflies, and 'bots: Integrating engineering and robotics into early childhood curricula", in English, L. and T. Moore (eds.), *Early Engineering Learning, Early Mathematics Learning and Development*, Springer, Singapore, https://doi.org/10.1007/978-981-10-8621-2_11. [164]

Elkin, M., A. Sullivan and M. Bers (2016), "Programming with the KIBO Robotics Kit in Preschool Classrooms", *Computers in the Schools*, Vol. 33/3, pp. 169-186, https://doi.org/10.1080/07380569.2016.1216251. [97]

Erickson, F. (1985), *Qualitative Methods in Research on Teaching*, Michigan State University, Institute for Research on Teaching, East Lansing. [74]

Faulkner, K. (2015), *Coding Across the Curriculum Resource Review*, http://dx.doi.org/10.13140/RG.2.1.2293.1929. [185]

Fayer, S., A. Lacey and A. Watson (2017), "STEM occupations: Past, present, and future", *US Bureau of Labor Statistics: Spotlight on Statistics*, Vol. 1, pp. 1-35. [5]

Fedorenko, E. et al. (2019), "The Language of Programming: A Cognitive Perspective", *Trends in Cognitive Sciences*, Vol. 23/7, pp. 525-528, https://doi.org/10.1016/j.tics.2019.04.010. [118]

Fisher, A. and J. Margolis (2003), "Unlocking the clubhouse", *ACM SIGCSE Bulletin*, Vol. 35/1, https://doi.org/10.1145/792548.611896. [112]

Flannery, L. et al. (2013), "Designing ScratchJr", *Proceedings of the 12th International Conference on Interaction Design and Children*, https://doi.org/10.1145/2485760.2485785. [57]

Folk, M. (1981), "Review of "Mindstorms: Children, Computers, and Powerful Ideas by Seymour Papert", Basic Books: New York, 1980", *ACM SIGCUE Outlook*, Vol. 15/1, pp. 23-24, https://doi.org/10.1145/1045071.1045074. [138]

Fowler, B. and E. Vegas (2021), *How England Implemented its Computer Science Education Program*, Center for Universal Education, Brookings, Washington DC, https://www.brookings.edu/wp-content/uploads/2021/01/How-England-implemented-its-computer-science-education-program.pdf (accessed on 17 May 2022). [39]

Fraillon, J. et al. (2020), *Preparing for Life in a Digital World: IEA International Computer and Information Literacy Study 2018 International Report*, Springer, Cham, https://doi.org/10.1007/978-3-030-38781-5. [23]

Fromberg, D. and L. Williams (eds.) (1992), "Perspectives on children", in *Encyclopedia of Early Childhood Education*, Routledge, Oxfordshire, https://doi.org/10.4324/9780203813546. [144]

Fromberg, D. (1990), "Play issues in early childhood education", in Seefeldt, C. (ed.), *Continuing Issues in Early Childhood Education*, Merrill, Columbus, OH, http://Back-to-Basics: Play in Early Childhood (studylib.net). [146]

Frost, J. (1992), *Play and Playscapes*, Delmar Publishers, Albany, NY. [145]

Gadanidis, G. (2015), "Coding as a Trojan horse for mathematics education reform", *Journal of Computers in Mathematics and Science Teaching*, Vol. 34/2, pp. 155-173. [126]

Garvey, C. (1977), *Play*, Harvard University Press, Cambridge, MA. [143]

Goode, J. and J. Margolis (2011), "Exploring computer science", *ACM Transactions on Computing Education*, Vol. 11/2, pp. 1-16, https://doi.org/10.1145/1993069.1993076. [184]

Google/Gallup (2015), *Searching for Computer Science: Access and Barriers in U.S. K-12 Education*, https://services.google.com/fh/files/misc/searching-for-computer-science_report.pdf (accessed on 17 May 2022). [41]

Gouvernment of Alberta (2013), *Learning and Technology Policy Framework*, http://learning-and-technology-policy-framework-web.pdf (alberta.ca). [192]

Gouvernment of Newfoundland and Labrador (2015), *Completely Kindergarten*, https://www.gov.nl.ca/education/files/k12_curriculum_guides_completely_kinder_9.-section-5-curriculum-framework-final.pdf. [189]

Govind, M. (2019), "Families that code together learn together: Exploring family-oriented programming in early childhoo with ScratchJr and KIBO Robotics", unpublished master's thesis, Tufts University, https://sites.tufts.edu/devtech/files/2019/07/Madhu-Govind_MA-Thesis_v5final.pdf. [239]

Govind, M. and M. Bers (2021), "Assessing robotics skills in early childhood: Development and testing of a tool for evaluating children's projects", *Journal of Research in STEM Education*, Vol. 7/1, pp. 47-68, https://doi.org/10.51355/jstem.2021.102. [223]

Govind, M. and M. Bers (2020), "Family Coding Days: Engaging Children and Parents in Creative Coding and Robotics", in Kalir, J. and D. Filipiak (eds.), *Proceedings of the 2020 Connected Learning Summit*, Carnegie Mellon University: - ETC Press, Pittsburgh, PA, https://2020.connectedlearningsummit.org/proceedings/. [240]

Granovskiy, B. (2018), *Science, Technology, Engineering, and Mathematics (STEM) Education: An Overview*, Congressional Research Service, Washington, DC, https://crsreports.congress.gov/product/pdf/R/R45223/4 (accessed on 17 May 2022). [85]

Grover, S. and R. Pea (2013), "Computational Thinking in K–12", *Educational Researcher*, Vol. 42/1, pp. 38-43, https://doi.org/10.3102/0013189x12463051. [12]

Gunther, C. (2017), "Informatik entdecken – mit und ohne Computer [Discover computer science – with and without a computer]", in Diethelm, I. (ed.), *Informatische Bildung zum Verstehen und Gestalten der digitalen Welt [Informatics education to understand and shape the digital world]*, Gesellschaft für Informatik e.V.(GI), Bonn, https://dl.gi.de/handle/20.500.12116/4347. [201]

Guzdial, M. (2008), "Education: Paving the way for computational thinking", *Communications of the ACM*, Vol. 51/8, pp. 25-27, https://doi.org/10.1145/1378704.1378713. [13]

Guzdial, M. and B. Morrison (2016), "Growing computer science education into a STEM education discipline", *Communications of the ACM*, Vol. 59/11, pp. 31-33, https://doi.org/10.1145/3000612. [115]

Hambrusch, S. et al. (2009), "A multidisciplinary approach towards computational thinking for science majors", *ACM SIGCSE Bulletin*, Vol. 41/1, pp. 183-187, https://doi.org/10.1145/1539024.1508931. [102]

Hassenfeld, Z. and M. Bers (2020), "Debugging the Writing Process: Lessons From a Comparison of Students' Coding and Writing Practices", *The Reading Teacher*, Vol. 73/6, pp. 735-746, https://doi.org/10.1002/trtr.1885. [58]

Hassenfeld, Z. et al. (2020), "If You Can Program, You Can Write: Learning Introductory Programming Across Literacy Levels", *Journal of Information Technology Education: Research*, Vol. 19, pp. 065-085, https://doi.org/10.28945/4509. [117]

Hazzan, O. (2008), "Reflections on teaching abstraction and other soft ideas", *ACM SIGCSE Bulletin*, Vol. 40/2, pp. 40-43, https://doi.org/10.1145/1383602.1383631. [130]

Heckman, J. and D. Masterov (2007), *The Productivity Argument for Investing in Young Children*, National Bureau of Economic Research, Cambridge, MA, https://doi.org/10.3386/w13016. [27]

Henderson, L. (2007), "Theorizing a Multiple Cultures Instructional Design Model for E-Learning and E-Teaching", in *Globalized E-Learning Cultural Challenges*, IGI Global, https://doi.org/10.4018/978-1-59904-301-2.ch008. [103]

Hermans, F. and E. Aivaloglou (2017), "To Scratch or not to Scratch?", *Proceedings of the 12th Workshop on Primary and Secondary Computing Education*, https://doi.org/10.1145/3137065.3137072. [175]

Honey, M. and D. Kanter (eds.) (2013), *Design, Make, Play: Growing the next generation of STEM innovators*, Routledge, New York, NY, https://www.routledge.com/Design-Make-Play-Growing-the-Next-Generation-of-STEM-Innovators/Honey/p/book/9780415539203. [227]

Horn, M. and M. Bers (2019), "Tangible computing", in Robins, A. and S. Fincher (eds.), *The Cambridge Handbook of Computing Education Research*, Cambridge University Press, Cambridge, https://doi.org/10.1017/9781108654555.023. [162]

Horn, M., R. Crouser and M. Bers (2011), "Tangible interaction and learning: the case for a hybrid approach", *Personal and Ubiquitous Computing*, Vol. 16/4, pp. 379-389, https://doi.org/10.1007/s00779-011-0404-2. [158]

Howes, C. (1987), "Social competence with peers in young children: Developmental sequences", *Developmental Review*, Vol. 7/3, pp. 252-272, https://doi.org/10.1016/0273-2297(87)90014-1. [71]

Hsu, T., S. Chang and Y. Hung (2018), "How to learn and how to teach computational thinking: Suggestions based on a review of the literature", *Computers & Education*, Vol. 126, pp. 296-310, https://doi.org/10.1016/j.compedu.2018.07.004. [45]

Huang, W. and C. Looi (2020), "A critical review of literature on "unplugged" pedagogies in K-12 computer science and computational thinking education", *Computer Science Education*, Vol. 31/1, pp. 83-111, https://doi.org/10.1080/08993408.2020.1789411. [178]

Hu, F. et al. (2015), "Strawbies: Explorations in tangible programming", in *Proceedings of the 14th International Conference on Interaction Design and Children*, Association for Computing Machinery, New York, NY, https://doi.org/10.1145/2771839.2771866. [155]

Ito, M. et al. (2013), *Connected Learning: An Agenda for Research and Design*, Digital Media and Learning Research Hub, Irvine, CA. [242]

Iwata, M. et al. (2020), "Exploring potentials and challenges to develop twenty-first century skills and computational thinking in K-12 Maker Education", *Frontiers in Education*, Vol. 5, https://doi.org/10.3389/feduc.2020.00087. [234]

Jones, N. (2016), "Digital technology to be added to education curriculum", *NZ Herald*, http://www.nzherald.co.nz/nz/news/article.cfm?c_id=1&objectid=11668961. [196]

K-12 Computer Science Framework Steering Committee (2016), *K–12 Computer Science Framework*, https://k12cs.org/wp-content/uploads/2016/09/K%E2%80%9312-Computer-Science-Framework.pdf (accessed on 17 May 2022). [40]

Kafai, Y., C. Proctor and D. Lui (2020), "From theory bias to theory dialogue", *ACM Inroads*, Vol. 11/1, pp. 44-53, https://doi.org/10.1145/3381887. [47]

Kalelioglu, F. and Y. Gulbahar (2014), "The Effects of Teaching Programming via Scratch on Problem Solving Skills: A Discussion from Learners' Perspective", *Informatics in Education*, Vol. 13/1, pp. 33-50, https://doi.org/10.15388/infedu.2014.03. [56]

Karpiński, Z., G. Di Pietro and F. Biagi (2021), "Computational thinking, socioeconomic gaps, and policy implications", *IEA Compass: Briefs in Education No. 12*, https://www.iea.nl/publications/series-journals/iea-compass-briefs-education-series/january-2021-computational. [246]

Kazakoff, E. and M. Bers (2014), "Put Your Robot in, Put Your Robot out: Sequencing through Programming Robots in Early Childhood", *Journal of Educational Computing Research*, Vol. 50/4, pp. 553-573, https://doi.org/10.2190/ec.50.4.f. [60]

Kazakoff, E., A. Sullivan and M. Bers (2012), "The Effect of a Classroom-Based Intensive Robotics and Programming Workshop on Sequencing Ability in Early Childhood", *Early Childhood Education Journal*, Vol. 41/4, pp. 245-255, https://doi.org/10.1007/s10643-012-0554-5. [61]

Kim, K. and J. Lee (2016), "Analysis of the effectiveness of computational thinking-based programming learning", *The Journal of Korean association of computer education*, Vol. 19/1, pp. 27-39. [21]

Kimmons, R., C. Graham and R. West (2020), "The PICRAT model for technology Integration in teacher preparation", *Contemporary Issues in Technology and Teacher Education*, Vol. 20/1, https://citejournal.org/volume-20/issue-1-20/general/the-picrat-model-for-technology-integration-in-teacher-preparation. [214]

Kirkman, G. (ed.) (2002), *Rethinking learning in the digital age*, Oxford University Press. [139]

Knight, V., J. Wright and A. DeFreese (2019), "Teaching Robotics Coding to a Student with ASD and Severe Problem Behavior", *Journal of Autism and Developmental Disorders*, Vol. 49/6, pp. 2632-2636, https://doi.org/10.1007/s10803-019-03888-3. [265]

Koehler, M. and P. Mishra (2009), "What is techlological pedagogical content knowledge?", *Contemporary Issues in Technology and Teacher Education*, Vol. 9/1, https://citejournal.org/volume-9/issue-1-09/general/what-is-technological-pedagogicalcontent-knowledge. [212]

Kong, S. (2016), "A framework of curriculum design for computational thinking development in K-12 education", *Journal of Computers in Education*, Vol. 3/4, pp. 377-394, https://doi.org/10.1007/s40692-016-0076-z. [83]

Kotsopoulos, D. et al. (2019), "Mathematical or Computational Thinking? An Early Years Perspective", in *Mathematical Learning and Cognition in Early Childhood*, Springer International Publishing, Cham, https://doi.org/10.1007/978-3-030-12895-1_6. [128]

Kramer, J. (2007), "Is abstraction the key to computing?", *Communications of the ACM*, Vol. 50/4, pp. 36-42, https://doi.org/10.1145/1232743.1232745. [129]

Kuhn, D., S. Nash and L. Brucken (1978), "Sex Role Concepts of Two- and Three-Year-Olds", *Child Development*, Vol. 49/2, p. 445, https://doi.org/10.2307/1128709. [254]

Kulju, P., R. Kupiainen and M. Pienimaki (2020), *Raportti luokanopettajien käsityksistä monilukutaidosta 2019 [Report on class teachers' perceptions of multi-literacy 2019]*, National Audiovisual Institute, Helsinki, https://trepo.tuni.fi/bitstream/handle/10024/123472/978-952-03-1762-1.pdf?sequence=2&isAllowed=y. [200]

Lee, I. et al. (2011), "Computational thinking for youth in practice", *ACM Inroads*, Vol. 2/1, pp. 32-37, https://doi.org/10.1145/1929887.1929902. [19]

Lee, K., A. Sullivan and M. Bers (2013), "Collaboration by Design: Using Robotics to Foster Social Interaction in Kindergarten", *Computers in the Schools*, Vol. 30/3, pp. 271-281, https://doi.org/10.1080/07380569.2013.805676. [78]

Lehrl, S. et al. (2021), "The Home Learning Environment in the Digital Age—Associations Between Self-Reported "Analog" and "Digital" Home Learning Environment and Children's Socio-Emotional and Academic Outcomes", *Frontiers in Psychology*, Vol. 12, https://doi.org/10.3389/fpsyg.2021.592513. [79]

Leidl, K., M. Bers and C. Mihm (2017), "Programming with ScratchJr: A review of the first year of user analytics", in Kong, S., J. Sheldon and R. Li (eds.), *Conference Proceedings of International Conference on Computational Thinking Education 2017*, The Education University of Hong Kong, Hong Kong, https://ase.tufts.edu/devtech/publications/Leidl_Bers_Mihm_ScratchJrAnalyticsHongKong.pdf. [51]

Lerner, R. and L. Steinberg (eds.) (2009), *Handbook of Adolescent Psychology*, John Wiley & Sons, Inc., Hoboken, NJ, USA, https://doi.org/10.1002/9780470479193. [30]

Levinson, T., L. Hunt and Z. Hassenfeld (2021), "Including Students With Disabilities in the Coding Classroom", in *Teaching Computational Thinking and Coding to Young Children, Advances in Early Childhood and K-12 Education*, IGI Global, https://doi.org/10.4018/978-1-7998-7308-2.ch012. [264]

Lye, S. and J. Koh (2014), "Review on teaching and learning of computational thinking through programming: What is next for K-12?", *Computers in Human Behavior*, Vol. 41, pp. 51-61, https://doi.org/10.1016/j.chb.2014.09.012. [24]

Magnuson, B. (2010), "Building blocks for mobile games: A multiplayer framework for app inventor for Android", doctoral dissertation, Massachusetts Institute of Technology, Dept. of Electrical Engineering and Computer Science, https://dspace.mit.edu/handle/1721.1/61253. [219]

Maguth, B. (2012), "Investigating Student Use of Technology for Engaged Citizenship in A Global Age", *Education Sciences*, Vol. 2/2, pp. 57-76, https://doi.org/10.3390/educsci2020057. [123]

Maiorana, F., D. Giordano and R. Morelli (2015), "Quizly: A live coding assessment platform for App Inventor", *2015 IEEE Blocks and Beyond Workshop (Blocks and Beyond)*, https://doi.org/10.1109/blocks.2015.7368995. [218]

Manches, A. and S. Price (2011), "Designing learning representations around physical manipulation", *Proceedings of the 10th International Conference on Interaction Design and Children - IDC '11*, https://doi.org/10.1145/1999030.1999040. [151]

Margolis, J. (2017), *Stuck in the Shallow End, updated edition: Education, Race, and Computing*, MIT Press. [113]

Marino, M. et al. (2013), "UDL in the Middle School Science Classroom", *Learning Disability Quarterly*, Vol. 37/2, pp. 87-99, https://doi.org/10.1177/0731948713503963. [271]

Markert, L. (1996), "Gender related success in science and technology", *The Journal of Technology Studies*, Vol. 22/2, pp. 21-29, https://www.jstor.org/stable/43604473. [243]

Martin, L. (2015), "The Promise of the Maker Movement for Education", *Journal of Pre-College Engineering Education Research (J-PEER)*, Vol. 5/1, https://doi.org/10.7771/2157-9288.1099. [228]

McClure, E. (2017), *STEM Starts Early*, http://Joan Ganz Cooney Center - STEM Starts Early: Grounding Science, Technology, Engineering, and Math Education in Early Childhood. [211]

McElwain, N. and B. Volling (2005), "Preschool children's interactions with friends and older siblings: relationship specificity and joint contributions to problem behavior.", *Journal of Family Psychology*, Vol. 19/4, pp. 486-496, https://doi.org/10.1037/0893-3200.19.4.486. [76]

McGlynn-Stewart, M. et al. (2019), "Open-Ended Apps in Kindergarten: Identity Exploration Through Digital Role-Play", *Language and Literacy*, Vol. 20/4, pp. 40-54, https://doi.org/10.20360/langandlit29439. [46]

McLaren, P. (2009), "Women and minorities in science, technology, engineering and mathematics: Upping the numbers. Edited by Ronald J. Burke and Mary C. Mattis (2007) Cheltenham, UK: Edward Elgar Publishing Limited, 379pp. ISBN: 978-1845428884", *Canadian Journal of Administrative Sciences / Revue Canadienne des Sciences de l'Administration*, Vol. 26/2, pp. 170-171, https://doi.org/10.1002/cjas.99. [259]

Metin, S. (2020), "Activity-based unplugged coding during the preschool period", *International Journal of Technology and Design Education*, pp. 1 - 17, http://Activity-based unplugged coding during the preschool period. International Journal of Technology and Design Education, 1-17. - Google Search. [176]

Ministère de l'Éducation et de l'Enseignement (2018), *Plan d'action numérique*, http://Plan d'action numérique en éducation et en enseignement supérieur | Ministère de l'Éducation et Ministère de l'Enseignement supérieur (gouv.qc.ca). [190]

Moomaw, S. and J. Davis (2010), "STEM comes to Preschool", *YC Young Children*, Vol. 65/5, pp. 12-14, 16-18. [90]

Moreno-León, J. and G. Robles (2015), "Dr. Scratch", *Proceedings of the Workshop in Primary and Secondary Computing Education*, https://doi.org/10.1145/2818314.2818338. [220]

Muñoz, R. et al. (2018), "Developing Computational Thinking Skills in Adolescents With Autism Spectrum Disorder Through Digital Game Programming", *IEEE Access*, Vol. 6, pp. 63880-63889, https://doi.org/10.1109/access.2018.2877417. [266]

Mussen, P. and L. Carmichael (eds.) (1983), *Peer relations*, Wiley. [70]

Myers, E. (2021), "The Role of Executive Function and Self-Regulation in the Development of Computational Thinking", in *Teaching Computational Thinking and Coding to Young Children, Advances in Early Childhood and K-12 Education*, IGI Global, https://doi.org/10.4018/978-1-7998-7308-2.ch004. [69]

NAEYC (2020), "Developmentally Appropriate Practice", *Position Statement Adopted by the NAEYC National Governing Board April 2020*, https://www.naeyc.org/sites/default/files/globally-shared/downloads/PDFs/resources/position-statements/dap-statement_0.pdf (accessed on November 2021). [210]

NAEYC and Fred Rogers Center (2012), "Technology and Interactive Media as Tools in Early Childhood Programs Serving Children from Birth through Age 8", *A joint position statement issued by the National Association for the Education of Young Children and the Fred Rogers Center for Early Learning and Children's Media at Saint Vincent College*, https://www.naeyc.org/resources/topics/technology-and-media/resources. [235]

National Center for Education Statistics (2022), *Students With Disabilities: Condition of Education*, U.S. Department of Education, Institute of Education Sciences, https://nces.ed.gov/programs/coe/indicator/cgg (accessed on 17 May 2022). [261]

National Center for Science and Engineering Statistics (2017), "Women, Minorities, and Persons with Disabilities in Science and Engineering", *Special Report NSF 17-310*, National Science Foundation, Arlington, VA, https://www.nsf.gov/statistics/2017/nsf17310/. [250]

National Research Council (2011), *Successful K-12 STEM Education: Identifying Effective Approaches in Science, Technology, Engineering, and Mathematics*, The National Academies Press, https://doi.org/10.17226/13158. [244]

National Research Council (2010), *Report of a Workshop on The Scope and Nature of Computational Thinking*, National Academies Press, Washington, D.C., https://doi.org/10.17226/12840. [14]

National Research Council (2001), *Eager to Learn: Educating Our Preschoolers*, National Academies Press, Washington, D.C., https://doi.org/10.17226/9745. [28]

New South Wales Department of Education (2019), *Coding and Computational thinking: What is the Evidence?*, 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 (accessed on 17 May 2022). [38]

OECD (2017), *The Pursuit of Gender Equality: An Uphill Battle*, OECD Publishing, Paris, https://dx.doi.org/10.1787/9789264281318-en. [251]

Ottenbreit-Leftwich, A. and A. Yadav (eds.) (2021), *Integration of computational thinking into English language arts*, ACM and the Robin Hood Learning + Technology Fund. [125]

Papert, S. (2000), "What's the big idea? Toward a pedagogy of idea power", *IBM Systems Journal*, Vol. 39/3.4, pp. 720-729, https://doi.org/10.1147/sj.393.0720. [273]

Papert, S. and I. Harel (1991), "Situating constructionism", in *Constructionism*, Ablex Publishing Corporation, Norwood, NJ, http://www.papert.org/articles/SituatingConstructionism.html. [229]

Pea, R. and D. Kurland (1984), "On the cognitive effects of learning computer programming", *New Ideas in Psychology*, Vol. 2/2, pp. 137-168, https://doi.org/10.1016/0732-118x(84)90018-7. [64]

Pellegrini, A. and P. Smith (1998), "Physical Activity Play: The Nature and Function of a Neglected Aspect of Play", *Child Development*, Vol. 69/3, pp. 577-598, https://doi.org/10.1111/j.1467-8624.1998.tb06226.x. [75]

Phillips, R. and B. Brooks (2017), *The Hour of Code: Impact on Attitudes Towards and Self-Efficacy with Computer Science*, https://code.org/files/ HourOfCodeImpactStudy_Jan2017.pdf. [150]

Pivetti, M. et al. (2020), "Educational Robotics for children with neurodevelopmental disorders: A systematic review", *Heliyon*, Vol. 6/10, p. e05160, https://doi.org/10.1016/j.heliyon.2020.e05160. [267]

Portelance, D., A. Strawhacker and M. Bers (2015), "Constructing the ScratchJr programming language in the early childhood classroom", *International Journal of Technology and Design Education*, Vol. 26/4, pp. 489-504, https://doi.org/10.1007/s10798-015-9325-0. [52]

Pretz, K. (2014), *Computer science classes for kids becoming mandatory*, The IEEE News Source, http://BeeBots and Tiny Tots - Learning & Technology Library (LearnTechLib). [195]

Przybylski, A. and N. Weinstein (2017), "Digital Screen Time Limits and Young Children's Psychological Well-Being: Evidence From a Population-Based Study", *Child Development*, Vol. 90/1, https://doi.org/10.1111/cdev.13007. [248]

Public Broadcasting Service (2020), *CPB and PBS Awarded Ready To Learn Grant from the U.S. Department of Education*, http://CPB and PBS Awarded Ready To Learn Grant from the U.S. Department of Education. [156]

Puentedura, R. (2013), *SAMR: Moving from Enhancement to Transformation*, http://www.hippasus.com/rrpweblog/archives/2013/05/29/SAMREnhancementToTransformation.pdf. [213]

Pugnali, A., A. Sullivan and M. Umashi Bers (2017), "The Impact of User Interface on Young Children's Computational Thinking", *Journal of Information Technology Education: Innovations in Practice*, Vol. 16, pp. 171-193, https://doi.org/10.28945/3768. [152]

Rambally, G. (2017), "Integrating Computational Thinking in Discrete Structures", in *Emerging Research, Practice, and Policy on Computational Thinking*, Springer International Publishing, Cham, https://doi.org/10.1007/978-3-319-52691-1_7. [127]

Rappolt-Schlichtmann, G. et al. (2013), "Universal Design for Learning and elementary school science: Exploring the efficacy, use, and perceptions of a web-based science notebook.", *Journal of Educational Psychology*, Vol. 105/4, pp. 1210-1225, https://doi.org/10.1037/a0033217. [272]

Redmond, P. et al. (2021), "Primary teachers' self-assessment of their confidence in implementing digital technologies curriculum", *Educational Technology Research and Development*, Vol. 69/5, pp. 2895-2915, https://doi.org/10.1007/s11423-021-10043-2. [204]

Relkin, E. (2021), "TechCheck", in *Teaching Computational Thinking and Coding to Young Children, Advances in Early Childhood and K-12 Education*, IGI Global, https://doi.org/10.4018/978-1-7998-7308-2.ch013. [226]

Relkin, E. (2018), *Assessing young children's computational thinking abilities*, http://hdl.handle.net/10427/015529. [15]

Relkin, E. and M. Bers (2021), *TechCheck-K: A Measure of Computational Thinking for Kindergarten Children*. [33]

Relkin, E. and M. Bers (2020), *Exploring the Relationship Between Coding, Computational Thinking and Problem Solving in Early Elementary School Students*. [25]

Relkin, E. and M. Bers (2019), "Designing an Assessment of Computational Thinking Abilities for Young Children", in *STEM in Early Childhood Education*, Routledge, https://doi.org/10.4324/9780429453755-5. [16]

Relkin, E., L. de Ruiter and M. Bers (2020), "TechCheck: Development and Validation of an Unplugged Assessment of Computational Thinking in Early Childhood Education", *Journal of Science Education and Technology*, Vol. 29/4, pp. 482-498, https://doi.org/10.1007/s10956-020-09831-x. [225]

Relkin, E. et al. (2020), "How Parents Support Children's Informal Learning Experiences with Robots", *Journal of Research in STEM Education*, Vol. 6/1, pp. 39-51, https://doi.org/10.51355/jstem.2020.87. [241]

Relkin, E. and A. Strawhacker (2021), "Unplugged Learning", in *Teaching Computational Thinking and Coding to Young Children, Advances in Early Childhood and K-12 Education*, IGI Global, https://doi.org/10.4018/978-1-7998-7308-2.ch003. [171]

Resnick, M. (2006), "Computer as Paintbrush: Technology, Play, and the Creative Society", in *Play = Learning*, Oxford University Press, https://doi.org/10.1093/acprof:oso/9780195304381.003.0010. [140]

Resnick, M. (1998), "Technologies for Lifelong Kindergarten", *Educational Technology Research and Development*, Vol. 46/4, pp. 43-55, http://www.jstor.org/stable/30220216. [62]

Resnick, M. et al. (2009), "Scratch", *Communications of the ACM*, Vol. 52/11, pp. 60-67, https://doi.org/10.1145/1592761.1592779. [142]

Rich, K. et al. (2019), "Synergies and differences in mathematical and computational thinking: implications for integrated instruction", *Interactive Learning Environments*, Vol. 28/3, pp. 272-283, https://doi.org/10.1080/10494820.2019.1612445. [133]

Rich, K., A. Yadav and R. Larimore (2020), "Teacher implementation profiles for integrating computational thinking into elementary mathematics and science instruction", *Education and Information Technologies*, Vol. 25/4, pp. 3161-3188, https://doi.org/10.1007/s10639-020-10115-5. [216]

Rich, K., A. Yadav and C. Schwarz (2019), "Computational Thinking, Mathematics, and Science: Elementary Teachers' Perspectives on Integration", *Journal of Technology and Teacher Education*, Vol. 27/2, pp. 165-295. [131]

Rideout, V. (2014), *Learning at Home: Families'Educational Media Use in America*, The Joan Ganz Cooney Center at Sesame Workshop, https://eric.ed.gov/?id=ED555586. [236]

Rodriguez, B. et al. (2017), "Assessing Computational Thinking in CS Unplugged Activities", *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*, https://doi.org/10.1145/3017680.3017779. [169]

Román-González, M., J. Moreno-León and G. Robles (2019), "Combining Assessment Tools for a Comprehensive Evaluation of Computational Thinking Interventions", in *Computational Thinking Education*, Springer Singapore, Singapore, https://doi.org/10.1007/978-981-13-6528-7_6. [48]

Rose, D. and A. Meyer (2002), *Teaching Every Student in the Digital Age: Universal Design for Learning*, Association for Supervision and Curriculum Development, Alexandria, VA, https://www.cast.org/products-services/resources/2002/universal-design-learning-udl-teaching-every-student-rose. [270]

Rubin, A. and R. Nemirovsky (1991), "Cars, computers, and air pumps: thoughts on the roles of physical and computer models in learning the central concepts of calculus", in Underbill, R. (ed.), *Proceedings of the 13th Annual Meeting, North American Chapter of the International Group for the Psychology of Mathematics Education*, Blacksburg, VA. [104]

Russ, S. (2003), *Play in Child Development and Psychotherapy*, Routledge, https://doi.org/10.4324/9781410609397. [147]

Ryoo, J. et al. (2013), "Democratizing computer science knowledge: transforming the face of computer science through public high school education", *Learning, Media and Technology*, Vol. 38/2, pp. 161-181, https://doi.org/10.1080/17439884.2013.756514. [114]

Saxena, C., H. Baber and P. Kumar (2020), "Examining the Moderating Effect of Perceived Benefits of Maintaining Social Distance on E-learning Quality During COVID-19 Pandemic", *Journal of Educational Technology Systems*, Vol. 49/4, pp. 532-554, https://doi.org/10.1177/0047239520977798. [249]

Scherer, R., F. Siddiq and B. Sánchez Viveros (2019), "The cognitive benefits of learning computer programming: A meta-analysis of transfer effects.", *Journal of Educational Psychology*, Vol. 111/5, pp. 764-792, https://doi.org/10.1037/edu0000314. [116]

Schofield, E., M. Erlinger and Z. Dodds (2014), "MyCS", *Proceedings of the 45th ACM technical symposium on Computer science education*, https://doi.org/10.1145/2538862.2538901. [183]

Sengupta, P. et al. (2013), "Integrating computational thinking with K-12 science education using agent-based computation: A theoretical framework", *Education and Information Technologies*, Vol. 18/2, pp. 351-380, https://doi.org/10.1007/s10639-012-9240-x. [105]

Settle, A. et al. (2012), "Infusing computational thinking into the middle- and high-school curriculum", *Proceedings of the 17th ACM annual conference on Innovation and technology in computer science education - ITiCSE '12*, https://doi.org/10.1145/2325296.2325306. [106]

Settle, A., D. Goldberg and V. Barr (2013), "Beyond computer science", *Proceedings of the 18th ACM conference on Innovation and technology in computer science education - ITiCSE '13*, https://doi.org/10.1145/2462476.2462511. [107]

Sheridan, K. et al. (2014), "Learning in the Making: A Comparative Case Study of Three Makerspaces", *Harvard Educational Review*, Vol. 84/4, pp. 505-531, https://doi.org/10.17763/haer.84.4.brr34733723j648u. [231]

Shonkoff, J. and D. Phillips (2000), *From Neurons to Neighborhoods*, National Academies Press, Washington, D.C., https://doi.org/10.17226/9824. [29]

Shute, V., C. Sun and J. Asbell-Clarke (2017), "Demystifying computational thinking", *Educational Research Review*, Vol. 22, pp. 142-158, https://doi.org/10.1016/j.edurev.2017.09.003. [17]

Signorella, M., R. Bigler and L. Liben (1993), "Developmental Differences in Children′s Gender Schemata about Others: A Meta-analytic Review", *Developmental Review*, Vol. 13/2, pp. 147-183, https://doi.org/10.1006/drev.1993.1007. [255]

Silva, E., B. Dembogurski and G. Semaan (2021), "A Systematic Review of Computational Thinking in Early Ages", *ArXiv*, https://arxiv.org/pdf/2106.10275.pdf. [153]

Singer, J. and D. Singer (2005), "Preschoolers' Imaginative Play as Precursor of Narrative Consciousness", *Imagination, Cognition and Personality*, Vol. 25/2, pp. 97-117, https://doi.org/10.2190/0kqu-9a2v-yam2-xd8j. [148]

Sinno, S. and M. Killen (2009), "Moms at Work and Dads at Home: Children's Evaluations of Parental Roles", *Applied Developmental Science*, Vol. 13/1, pp. 16-29, https://doi.org/10.1080/10888690802606735. [256]

So, H., M. Jong and C. Liu (2019), "Computational Thinking Education in the Asian Pacific Region", *The Asia-Pacific Education Researcher*, Vol. 29/1, pp. 1-8, https://doi.org/10.1007/s40299-019-00494-w. [202]

Spanish Government Ministry of Education and Vocational Training (2021), *Escuela de Pensamiento computacional e Inteligencia Artificial [School of Computational Thinking and Artificial Intelligence]*, https://intef.es/tecnologia-educativa/pensamiento-computacional/. [198]

Spanish Government, Ministry of Education and Vocational Training (2018), *Programación, Robótica y Pensamiento Computacional en el Aula [Programming, robotics and Computational Thinking in the Classroom]*, Ministry of Education and Vocational Training, Madrid, https://code.intef.es/wp-content/uploads/2017/09/Fase-2-Informe-sobre-la-situaci%C3%B3n-en-Espa%C3%B1a-actualizado-y-propuesta-normativa-inf-y-prim.pdf. [199]

Spencer, S., C. Steele and D. Quinn (1999), "Stereotype Threat and Women's Math Performance", *Journal of Experimental Social Psychology*, Vol. 35/1, pp. 4-28, https://doi.org/10.1006/jesp.1998.1373. [253]

Steele, C. (1997), "A threat in the air: How stereotypes shape intellectual identity and performance.", *American Psychologist*, Vol. 52/6, pp. 613-629, https://doi.org/10.1037/0003-066x.52.6.613. [252]

Strawhacker, A. and M. Bers (2014), "ScratchJr: Computer Programming in Early Childhood Education as a Pathway to Academic Readiness and Success", *Poster session presented at the DR K-12 PI Meeting*, https://cadrek12.org/2014-dr-k-12-pi-meeting/posterhall. [59]

Strawhacker, A. et al. (2020), "The Role of Picture Books in supporting Young Children's Learning about Bioengineering [Poster session]", *SRCD 2020 Special Topic Meeting: Learning through Play and Imagination*, St. Louis, MO. [137]

Strawhacker, A., M. Lee and M. Bers (2017), "Teaching tools, teachers' rules: exploring the impact of teaching styles on young children's programming knowledge in ScratchJr", *International Journal of Technology and Design Education*, Vol. 28/2, pp. 347-376, https://doi.org/10.1007/s10798-017-9400-9. [53]

Strawhacker, A. et al. (2015), "ScratchJr demo: A coding language for kindergarten", *IDC'15: Proceedings of the 14th International Conference on Interaction Design and Children, Boston, MA, 21-24 June 2015*, https://doi.org/10.1145/2771839.2771867. [54]

Strawhacker, A. et al. (2020), "Debugging as Inquiry in Early Childhood: A case study using the CRISPEE prototype", *in Roundtable Session (Chair: K. Mills): Computational Thinking for Science Learning at the Annual Meeting of the American Educational Research Association (AERA)*, AERA. [136]

Strawhacker, A. et al. (2020), "Designing with Genes in Early Childhood: An exploratory user study of the tangible CRISPEE technology", *International Journal of Child-Computer Interaction*, Vol. 26, p. 100212, https://doi.org/10.1016/j.ijcci.2020.100212. [134]

Strawhacker, A. et al. (2020), "Young Children's Learning of Bioengineering with CRISPEE: a Developmentally Appropriate Tangible User Interface", *Journal of Science Education and Technology*, Vol. 29, pp. 319-339, https://doi.org/10.1007/s10956-020-09817-9. [135]

Sullivan, A. (2021), "Supporting Girls' Computational Thinking Skillsets", in *Teaching Computational Thinking and Coding to Young Children, Advances in Early Childhood and K-12 Education*, IGI Global, https://doi.org/10.4018/978-1-7998-7308-2.ch011. [257]

Sullivan, A. (2019), *Breaking the STEM stereotype: Reaching girls in early childhood*, Rowman & Littlefield Publishers. [35]

Sullivan, A. and M. Bers (2019), "Computer Science Education in Early Childhood: The Case of ScratchJr", *Journal of Information Technology Education: Innovations in Practice*, Vol. 18, pp. 113-138, https://doi.org/10.28945/4437. [44]

Sullivan, A. and M. Bers (2017), "Dancing robots: integrating art, music, and robotics in Singapore's early childhood centers", *International Journal of Technology and Design Education*, Vol. 28/2, pp. 325-346, https://doi.org/10.1007/s10798-017-9397-0. [99]

Sullivan, A., M. Bers and C. Mihm (2017), "Imagining, Playing, & Coding with KIBO: Using KIBO Robotics to Foster Computational Thinking in Young Children", *in the Proceedings of the International Conference on Computational Thinking Education, Education University of Hong Kong*. [65]

Sullivan, A., M. Elkin and M. Bers (2015), "KIBO robot demo", *Proceedings of the 14th International Conference on Interaction Design and Children*, https://doi.org/10.1145/2771839.2771868. [66]

Sullivan, A. and A. Strawhacker (2021), "Screen-Free STEAM: Low-Cost and Hands-on Approaches to Teaching Coding and Engineering to Young Children", in *Embedding STEAM in Early Childhood Education and Care*, Springer International Publishing, Cham, https://doi.org/10.1007/978-3-030-65624-9_5. [94]

Sullivan, A., A. Strawhacker and M. Bers (2017), "Dancing, Drawing, and Dramatic Robots: Integrating Robotics and the Arts to Teach Foundational STEAM Concepts to Young Children", in *Robotics in STEM Education*, Springer International Publishing, Cham, https://doi.org/10.1007/978-3-319-57786-9_10. [98]

Sullivan, A. and M. Umashi Bers (2016), "Girls, Boys, and Bots: Gender Differences in Young Children's Performance on Robotics and Programming Tasks", *Journal of Information Technology Education: Innovations in Practice*, Vol. 15, pp. 145-165, https://doi.org/10.28945/3547. [163]

Sun, L. et al. (2020), "STEM learning attitude predicts computational thinking skills among primary school students", *Journal of Computer Assisted Learning*, Vol. 37/2, pp. 346-358, https://doi.org/10.1111/jcal.12493. [31]

Svensson, A. (2000), "Computers in School: Socially Isolating or a Tool to Promote Collaboration?", *Journal of Educational Computing Research*, Vol. 22/4, pp. 437-453, https://doi.org/10.2190/30kt-1vlx-fhtm-rcd6. [73]

Taylor, M. (2018), "Computer Programming With Pre-K Through First-Grade Students With Intellectual Disabilities", *The Journal of Special Education*, Vol. 52/2, pp. 78-88, https://doi.org/10.1177/0022466918761120. [263]

Taylor, M., E. Vasquez and C. Donehower (2017), "Computer Programming with Early Elementary Students with Down Syndrome", *Journal of Special Education Technology*, Vol. 32/3, pp. 149-159, https://doi.org/10.1177/0162643417704439. [268]

The Royal Society (2021), "STEM sector must step up and end unacceptable disparities in Black staff and students academic progression and success", https://royalsociety.org/news/2021/03/stem-ethnicity-report/ (accessed on 17 May 2022). [42]

Thies, R. and J. Vahrenhold (2013), "On plugging "unplugged" into CS classes", *Proceeding of the 44th ACM technical symposium on Computer science education - SIGCSE '13*, https://doi.org/10.1145/2445196.2445303. [180]

Thies, R. and J. Vahrenhold (2012), "Reflections on outreach programs in CS classes", *Proceedings of the 43rd ACM technical symposium on Computer Science Education - SIGCSE '12*, https://doi.org/10.1145/2157136.2157281. [179]

Trevallion, D. (2014), "Connecting to Australia's first digital technology curriculum", *The Conversation*, http://theconversation.com/connecting-to-australias-first-digital-technology-curriculum-23507. (accessed on 17 May 2022). [197]

Tsortanidou, X., T. Daradoumis and E. Barberá (2021), "A K-6 computational thinking curricular framework: pedagogical implications for teaching practice", *Interactive Learning Environments*, pp. 1-21, https://doi.org/10.1080/10494820.2021.1986725. [84]

Tucker, A. (2003), *A model curriculum for k--12 computer science: Final report of the ACM K-12 task force curriculum committee*, Association for Computing Machinery (ACM), https://dl.acm.org/doi/book/10.1145/2593247. [7]

Unahalekhaka, A. and M. Bers (2022), "Clustering Young Children's Coding Project Scores with Machine Learning", *2022 IEEE Global Engineering Education Conference (EDUCON)*, https://doi.org/10.1109/educon52537.2022.9766579. [221]

Unahalekhaka, A. and M. Bers (2022), "Evaluating young children's creative coding: rubric development and testing for ScratchJr projects", *Education and Information Technologies*, https://doi.org/10.1007/s10639-021-10873-w. [222]

Unahalekhaka, A. and M. Govind (2021), "Examining Young Children's Computational Artifacts", in *Teaching Computational Thinking and Coding to Young Children, Advances in Early Childhood and K-12 Education*, IGI Global, https://doi.org/10.4018/978-1-7998-7308-2.ch014. [187]

Upadhyaya, B., M. McGill and A. Decker (2020), "A Longitudinal Analysis of K-12 Computing Education Research in the United States", *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, https://doi.org/10.1145/3328778.3366809. [173]

US Government National Science and Technology Council (2018), *Charting a Course for Success: America's Strategy for STEM Education*, US Government, Office of Science and Technology Policy, Washington D.C., https://www.energy.gov/sites/default/files/2019/05/f62/STEM-Education-Strategic-Plan-2018.pdf. [86]

Vee, A. (2017), *Coding Literacy*, The MIT Press, https://doi.org/10.7551/mitpress/10655.001.0001. [121]

Vee, A. (2013), "Understanding Computer Programming as a Literacy", *Literacy in Composition Studies*, Vol. 1/2, pp. 42-64, https://doi.org/10.21623/1.1.2.4. [120]

Wahlström, J. et al. (2000), "Differences between work methods and gender in computer mouse use", *Scandinavian Journal of Work, Environment & Health*, Vol. 26/5, pp. 390-397, https://doi.org/10.5271/sjweh.559. [160]

Walter-Herrmann, J. and C. Buching (eds.) (2013), *Digital fabrication and 'making'in education: The democratization of invention*, Transcript Publishers. [109]

Wan, Z., Y. Jiang and Y. Zhan (2020), "STEM Education in Early Childhood: A Review of Empirical Studies", *Early Education and Development*, Vol. 32/7, pp. 940-962, https://doi.org/10.1080/10409289.2020.1814986. [80]

Wartella, E. and N. Jennings (2000), "Children and Computers: New Technology. Old Concerns", *The Future of Children*, Vol. 10/2, p. 31, https://doi.org/10.2307/1602688. [77]

Watson, E. (2020), "STEM or STEAM?: the Critical Role of Arts in Technology Education (and the Critical Role of Art in Technology)", *Irish Journal of Academic Practice*, Vol. 8/1, https://arrow.tudublin.ie/cgi/viewcontent.cgi?article=1079&context=ijap (accessed on 17 May 2022). [92]

Weintrop, D. et al. (2015), "Defining Computational Thinking for Mathematics and Science Classrooms", *Journal of Science Education and Technology*, Vol. 25/1, pp. 127-147, https://doi.org/10.1007/s10956-015-9581-5. [108]

White House, US (2016), "Fact sheet: Advancing active STEM for our yongest learners", *The White House Office of the Press Secretary news release of 21 April 2016*, https://obamawhitehouse.archives.gov/the-press-office/2016/04/21/fact-sheet-advancing-active-stem-education-our-youngest-learners. [188]

Wing, J. (2011), "Research notebook: Computational thinking—What and why", *The link magazine* 6, pp. 20-23, http://www.cs.cmu.edu/link/research-notebook-computational-thinking-what-and-why. [2]

Wing, J. (2006), "Computational thinking", *Communications of the ACM*, Vol. 49/3, pp. 33-35, https://doi.org/10.1145/1118178.1118215. [1]

Wohl, B., B. Porter and S. Clinch (2015), "Teaching Computer Science to 5-7 year-olds", *Proceedings of the Workshop in Primary and Secondary Computing Education*, https://doi.org/10.1145/2818314.2818340. [177]

World Economic Forum (2020), *The Future of Jobs Report 2020*, Geneva: World Economic Forum., http://www.weforum.org/reports/the-future-of-jobs-report-2020/in-full. [6]

World Health Organisation (2019), *Guidelines on Physical Activity, Sedentary Behaviour and Sleep for Children under 5 Years of Age*, World Health Organisation, Geneva, https://apps.who.int/iris/handle/10665/311664. [166]

World Health Organization and World Bank (2011), *World report on disability 2011*, World Health Organization, https://apps.who.int/iris/handle/10665/44575. [260]

Wortham, S. (2009), *Early Childhood Curriculum: Developmental Bases for Learning and Teaching*, Pearson Merrill Prentice Hall, Upper Saddle River, NJ. [88]

Yadav, A. et al. (2016), "Expanding computer science education in schools: understanding teacher experiences and challenges", *Computer Science Education*, Vol. 26/4, pp. 235-254, https://doi.org/10.1080/08993408.2016.1257418. [205]

Yadav, A. et al. (2018), "Computational thinking in elementary classrooms: measuring teacher understanding of computational ideas for teaching science", *Computer Science Education*, Vol. 28/4, pp. 371-400, https://doi.org/10.1080/08993408.2018.1560550. [208]

Yakman, G. (2008), *STEAM Education: an Overview of Creating a Model of Integrative Education.*, https://www.researchgate.net/publication/327351326_STEAM_Education_an_overview_of_creating_a_model_of_integrative_education (accessed on 17 May 2022). [93]

Zhang, L. and J. Nouri (2019), "A systematic review of learning computational thinking through Scratch in K-9", *Computers & Education*, Vol. 141, p. 103607, https://doi.org/10.1016/j.compedu.2019.103607. [49]