Can. J. Sci. Math. Techn. Educ. (2023) 23:509–537
https://doi.org/10.1007/s42330-023-00293-y

UNIVERSITY OF TORONTO
OISE | ONTARIO INSTITUTE
FOR STUDIES IN EDUCATION

Check for updates

The Integration of Coding and Computer Science Concepts in Canadian K-8 Curriculum

Steven Floyd[1]

Accepted: 24 October 2023 / Published online: 7 December 2023
© Ontario Institute for Studies in Education (OISE) 2023

Abstract In this article, the relatively new phenomenon of integrating coding and computer science (CS) related concepts and skills into the K-8 grades is analysed through a comparative analysis of related provincial curriculum initiatives in Canada. First, provincial K-8 curricula that include coding and CS related concepts and skills are identified, as well as the placement of these components within the provincial policy documents. This is followed by a comparative analysis of stated aims and objectives of the curriculum components, and an analysis of the selected concepts and skills themselves. Throughout this analysis, context is provided by theory in the field, as well as the general approaches from jurisdictions outside of Canada, which have been found in the literature. What results is a comparative analysis of this nascent curriculum topic as well as important insights for educators, policy makers, and researchers alike.

Résumé Dans cet article, par le biais d'un examen comparatif des initiatives provinciales connexes en matière de programmes d'études en vigueur au Canada, on analyse le phénomène relativement récent de l'intégration des compétences et des concepts liés à l'informatique dans les classes de la maternelle à la 8e année. On cerne tout d'abord les programmes provinciaux de la maternelle à la 8e année qui incluent le codage et les compétences et concepts qui y sont associés, ainsi que l'endroit où se retrouvent ces éléments dans les documents énonçant les politiques provinciales. Vient ensuite une analyse comparative des buts et objectifs déclarés des éléments faisant partie du programme, ainsi qu'un examen des compétences et des concepts choisis. Tout au long de cette analyse, le contexte provient de la théorie liée au domaine ainsi que des approches générales relevant de compétences situées hors du Canada trouvées dans la documentation. Il en résulte une analyse comparative de ce sujet particulier (à l'état naissant) du programme d'étude ainsi que des connaissances précieuses, autant pour les éducateurs, les décideurs politiques que pour les chercheurs.

Keywords Computer science · Coding · Computational thinking · Curriculum

✉ Steven Floyd
stevenpfloyd@gmail.com

[1] London, Canada

Introduction

Educational systems around the world have been undergoing reforms to ensure that their policies and practices adequately prepare students to meet the changing needs of life and work as school experiences do not align with the needs of a diverse, rapidly changing, and technologically sophisticated society (Milton, 2015). The integration of coding and related concepts in the K-8 grades has become a component of these reforms (Bocconi et al., 2016; Dagienė et al., 2019). Coding and associated computer science (CS) concepts can form the basis of lucrative, high-status, and flexible careers (Information and Communications Technology Council, 2017), but others argue that the integration of coding concepts and skills in the K-8 grades should be motivated by more than simply economic goals (Tissenbaum et al., 2021; Lee & Soep, 2023).

A number of studies analyzing curricula from a variety of educational jurisdictions have identified different goals and rationale for the integration of coding in the younger grades (Webb et al., 2015; Passey, 2017; Vogel et al., 2017; Hubweiser et al., 2015). In addition, the literature reveals a variety of theoretical perspectives (Kafai, 2016; diSessa, 2018; Resnick, 2018; Tissenbaun et al., 2019). These goals, rationale, and perspectives will be explored in the following two sections.

Arguments for Coding Curriculum in the Younger Grades

Before considering the placement of coding and related concepts and skills in K-8 provincial curricula, it is important to develop an understanding of the various goals associated with younger students programming a computer. Passey (2017) identifies six main reasons for the inclusion of CS curricula in the younger grades that include the economic argument, the organizational argument, the community argument, the educational argument, the learning argument, and the learner argument. Passey's (2017) economic argument is workforce centred, focusing on the idea that curriculum should support future economies and should support students in developing the skills needed to meet the needs of future careers. This argument is based on the idea that specific coding-related concepts and skills will be valuable for future careers. In contrast, Passey's organizational argument, while still connected to economic and workforce motivators, is broader and recognizes the potential of coding curriculum leading to collaboration and teamwork-related skills, which he states will also be in demand in future careers. Moving beyond the workplace, the community argument recognizes the need for general computing capabilities to support community groups and programmes, such as a supporting social bird watching and music groups or allowing older individuals leveraging technology to maintain communication and connections with others. The educational argument is focused on all individuals being provided with the opportunity to learn important digital skills that all citizens should have, and about understanding the coding and CS concepts that lay behind our ubiquitous technologies. Closely connected to the educational argument is the learning argument, which recognizes the associated problem solving, creativity, and logical thinking skills sometimes associated with coding and CS work. When discussing the learning argument, Passey introduces Seymour Papert's work on constructionism, which will be explored later in this article. Finally, Passey's learner argument puts the student at the centre of the curriculum, recognizing that students are often motivated and engaged when programming a computer, and young students should be provided with the opportunity to explore coding and CS concepts as a potential area of interest and focus.

In addition to Passey's six arguments, other works have identified differing goals and rationale for coding curriculum in the younger grades. These goals and rationale sometimes not only overlap with Passey's arguments, but also add insights and direction that Passey left out. Vogel et al. (2017) identified seven areas of impact present in arguments for universal CS education, including (1) economic and

workforce development; (2) equity and social justice; (3) competencies and literacies; (4) citizenship and civic life; (5) scientific, technological, and social innovation; (6) school improvement and reform; and (7) fun, fulfillment, and personal agency. While many of these share ideas from Passey's arguments, the equity and social justice perspective and the motivation for scientific and technological innovation perspective add new dimensions and considerations that Passey did not emphasize. Equity and social justice perspectives often relate to the need for citizens to be active and critical users of technology, and are associated with related concepts such as privacy or safety (Fluck et al., 2016), as well as equity issues surrounding gender equality, and underrepresented groups in CS education, or the CS field in general. Arguments surrounding scientific and technological innovation recognize coding and CS concepts as a critical component of a cross-curricular, science, technology, engineering, and mathematics (STEM) education.

Also left out of Passey's arguments and identified by Webb et al. (2015) are the cultural reasons for the inclusion of coding concepts and skills in curriculum. These cultural reasons are associated with empowerment, and the recognition of coding and CS concepts and skills as "enabling people to be the drivers of cultural change, rather than having change imposed by technological developments" (Webb et al., 2017, p. 446).

Table 1 outlines a general organization of recent arguments for the inclusion of coding concepts and skills in the curricula of the younger grades. Webb et al.'s (2015) broad categories are included first, then Passey's (2017) and Vogel et al.'s (2017) detailed areas of focus. Also included are the detailed categories of goals identified by Hubweiser et al. (2015). In "A Global Snapshot of Computer Science Education in K-12 Schools", Hubweisser et al. analyzed and summarized 14 articles, published in two special issues of *Computer Science Education in K-12 Schools*, that included information related to K-12 CS education from 12 countries or states from around the world. Through the analysis of these articles, the authors identified 19 categories of addressed goals, many of which fit into Webb et al.'s (2015) general categories, but add specificity and detail.

Theoretical Perspectives on Coding in the K-8 Grades

In addition to Seymour Papert's (1993) foundational work related to the Logo programming language and the learning theory of constructionism, a number of relatively recent theoretical approaches have been developed that relate to coding concepts and skills in the younger grades. These include Computational Thinking (Wing, 2006; Grover & Pea, 2013, 2018), Fluency (Resnick, 2018), Participation (Kafai, 2016), Literacy (diSessa, 2000, 2018), and Action (Tissenbaum et al., 2019). In combination with the arguments for coding in the K-12 grades (listed above in Table 1), an understanding of the similarities and differences of these theoretical approaches is important in order to inform analysis of coding curricula.

Constructionism

Constructionism arose from the work of Jean Piaget, with whom Papert had worked, and who articulated the theory of cognitive development called constructivism. Harel and Papert (1991) explain that the learning theory of constructionism can be over-simplified and thought of as "learning-by-making"; however, it is much more multifaceted than this, and has much deeper implications. Ames (2018) explains that both constructivism and constructionism focus on learning being an active process of constructing knowledge, and both support the idea that children learn new concepts by relating them to things that they already know. An important distinction between the two, however, is that constructionism includes the idea that this can happen felicitously when the learner is constructing something that others might see (Harel & Papert, 1991). A key goal of constructionism is "to respect children as creators, enable

Table 1 Recent arguments and goals for coding in the younger grades

Webb et al. (2015)	Passey (2017)	Vogel et al. (2017)	Hubweiser et al. (2015)
• Economic • Social • Cultural	• Economic argument • Organizational argument • Community argument • Educational argument • Learning argument • Learner argument	• Economic and workforce development • Equity and social justice • Competencies and literacies • Citizenship and civic life • Scientific, technological, and social innovation • School improvement and reform • Fun, fulfillment, and personal agency	• Digital literacy • Computational thinking • Problem solving • Understanding basic concepts of CS and it • Career preparation and choice • Support awareness of social, ethical, legal, and privacy issues and impact of CS • General education to participate in society responsibly • Prepare for university • Student development • Attract and motivate more female and male students • Create IT • Holistic view • Connecting to real-world contexts • Creative use of IT • Limits and risks of CS • Support communication about IT • Support mathematics and science • Apply IT in other subjects • Deeper knowledge of CS • Growth of knowledge society • Modern and relevant curriculum • Picture of CS and programming in society • Representing thinking processes • Rise and discover talent and attitude towards CS

them to engage in making meaning for themselves through construction, and to do this by democratizing access to the world's most creative and powerful tools" (Holbert et al., 2020).

Speaking specifically about mathematics education, the Harel and Papert (1991) indicate that having students work creative and powerful tool such as "cybernetic construction kits", which essentially combined Papert's Logo coding software with physical, robotics-like LEGO kits, changes the context of learning and holds the attention of students for much longer (Harel & Papert, 1991). While Papert acknowledged the construction of a public entity might not require a computer, it could be a soap-sculpture or even a knot-tying project, he does emphasize that the computer can serve as a Proteus of machines, taking on a thousand forms and serving a thousand functions (Papert, 1993). In this way, the computer can help relieve what he calls the potential poverty of a classroom culture, which might lack the needed resources and materials to support a wide range of learning opportunities for students. As a result, the computer played a central role in Papert's work with children, and his focus was always on the mind and the way in which technology could provide children with new possibilities for learning, thinking, and growing, both cognitively and emotionally (Papert, 1993).

More recently, studies have incorporated a constructionist framework as they investigated educational contexts in which a computer or physical robots were present. Khanlari (2013) concludes that student learning is improved when participants engaged in the development of a robot, which was a personally meaningful product, and Sullivan and Heffernan (2016) suggested that the constructionist learning affordances of computational manipulatives (e.g. immediate feedback, multiple modes of representation) may provide greater learning opportunities than alternative approaches. Additionally, Papavlasopoulou et al. (2019) suggest a need for more studies that use constructionism as a theoretical grounding. While these studies investigate broader STEM, twenty-first century learning, and specific coding skills, Papert often remained focused on the learning within the realm of mathematics.

A thorough description of Papert's views related to coding and mathematics can be found in his book *Mindstorms* (1993), where he describes a mechanical thinking process that students undergo when programming a computer (p. 27), and also describes a term called computational thinking (p. 182). Thirteen years after the release of *Mindstorms*, Wing (2006) used the term computational thinking, albeit in a different way, and captured the interest of educators and researchers in K-12 education from around the world (Grover & Pea, 2013).

Computational Thinking

Wing (2006) defines computational thinking (CT) as a "universally applicable attitude and skill set everyone, not just computer scientists, would be eager to learn and use" (p. 33). Wing identifies solving problems, system design, and understanding human behavior as key components of her definition of CT. She explains that CT is a fundamental skill that every human must know to function in society. In addition to being for everyone, everywhere, Wing states that CT involves conceptualization, rather than programming, and involves ideas, rather than artefacts. Her article was a call for the inclusion of CT not only in post-secondary programmes outside of CS, but also in pre-college education where younger students could be exposed to computational methods and models: "Computational thinking is a grand vision to guide CS educators, researchers, and practitioners as we act to change society's image of the field" (p. 35).

While most researchers agree on the profound impact that Wing's 2006 article had on the field of K-12 education (as of May 2023, this article had been cited 10,845 times), not all agree on the appropriateness of her definition, or on her suggestion that thinking like a computer scientist is a suitable goal for all students. Denning (2017) claims that recent attempts to make CT appealing to fields other than CS have led to "vague and confusing definitions of CT" (p. 33), and that Wing's definition lacks any mention of computational models, and incorrectly suggests that any sequence of steps constitutes

an algorithm. In "Computational Thinking: A Competency Whose Time Has Come", Grover and Pea (2018) describe Wing's definition as somewhat opaque. They attempt to rectify this concern by providing a specific, and much needed, list of CT concepts and practices that describe the type of thinking that computer scientists activate when engaged in problem solving. Grover and Pea's key CT concepts include logic and logical thinking, algorithms and algorithmic thinking, patterns and pattern recognition, abstraction and generalization, evaluation, and automation. Their key CT concepts include problem decomposition, creating computational artefacts, testing, and debugging, iterative refinement, and collaboration and creativity. Similarly, Brennan and Resnick (2012) gave more detail to CT by identifying and describing specific concepts, practices, and perspectives, while Resnick (2018) also describes an alternative theoretical approach that he terms computational fluency.

Computational Fluency

In "New Frameworks for Studying and Assessing the Development of Computational Thinking", Brennan and Resnick (2012) acknowledge the disagreements surrounding the components of CT, and the issues surrounding strategies for CT assessment. Like Grover and Pea (2018), Resnick and Brennan provide the specific detail that was lacking in Wing's original definition of CT, and introduce their own CT concepts, practices, and perspectives. These concepts, practices, and perspectives are listed in Table 2, along with Grover and Pea's concepts and practices.

In addition to the CT concepts, practices, and perspectives presented with Brennan, in 2018, Resnick also introduced his concept of computational fluency, which expands upon computation concepts and problem-solving strategies, in order to also include student's creativity and expression of digital tools

Table 2 Brennan and Resnick's (2012) CT concepts, practices, and perspectives and Grover and Pea's (2018) concepts and practices

Brennan and Resnick (2012)	Grover and Pea (2018)
Concepts that students engage in when developing coding projects: • Sequences; • Loops; • Parallelism; • Events; • Conditionals; • Operators; and • Data	Concepts: • Logic and logical thinking; • Algorithms and algorithmic thinking; • Patterns and pattern recognition; • Abstraction and generalization; and • Evaluation, and automation
Practices that describe the processes of construction that student engage in while developing coding projects: • Being incremental and iterative; • Testing and debugging; • Reusing and remixing; and • Abstracting and modularizing	Practices that outline approaches that computer scientists often use when they engage in computational problem solving: • Problem decomposition; • Creating computational artefacts; • Testing and debugging; • Iterative refinement; and • Collaboration and creativity
Perspectives that describe the evolving understanding that students exhibit about themselves, their relationship to others, and the technological world when developing coding projects: • Expressing; • Connecting; • And questioning	

(Resnick, 2018). While Resnick acknowledges the value of self-contained "coding puzzles" and their potential development of thinking skills, he argues that students should move towards developing a voice and an identity within the area of coding, and can do so by incorporating coding into their daily life, and by emphasizing the development of artefacts and projects (Resnick, 2018). This development of artefacts and projects connects closely to aspects of design and engineering that sometimes appear in curriculum, and computational fluency could serve as a valuable context for learning within these areas.

Computational Participation

Sharing Resnick's belief in the importance of students moving beyond coding puzzles to creating their own artefacts, Kafai (2016) goes one step further to highlight the importance of students being able to share coding projects that they have designed themselves, with others, moving beyond the tools, to focus on how the artefacts of coding can connect to community and context (Kafai, 2016). Kafai's computational participation recognizes the importance of digital technologies being used for functional, political, and personal reasons, and acknowledges coding as a participatory process that has a personal value, and value for sharing with others (Kafai 2016). "Computational thinking and programming are social, creative practices. They offer a context for making applications of significance for others, communities in which design, sharing, and collaboration with others are paramount" (Kafai, 2016, p. 26). Kafai describes some of the do-it-yourself coding tools available to students today to design, create, and share projects online, and identifies three new pathways that are afforded through these tools. The first pathway includes moving from simply building code to developing shareable applications, which puts the emphasis on putting newfound coding skills to use, rather than coding for the sake of coding. The second pathway includes moving from solitary coding to the development of communities, where coding languages and environments are enhanced by having online communities that connect users and provide audiences for projects. The final pathway includes having students remix existing projects, rather than beginning writing a program from scratch, which in the spirit of the open-source movement, allows for students to understand how projects can evolve and lead to innovative new contexts.

Computational Action

Computational action was first described by Tissenbaum et al. (2019) and, like Resnick's computational fluency and Kafai's computational participation, highlights the importance of the artefact being produced, and its potential influence outside of the individual student, or school context. Recognizing the impact that computing can have on the lives of the students and their communities, the authors present the two key dimensions of computational identity and computational empowerment as means to make computing more inclusive, motivating, and empowering. Computational action attempts to provide an alternative to the "fundamentals approach" that begins with a focus on coding or CT concepts and processes, by ensuring that students can immediately begin to code projects that connect to their lives, and that can help them develop a "critical consciousness of the role they can play in affecting their communities through computing and empower them to move beyond simply learning to code" (Tissenbaum et al., 2019, p. 34).

In order to support the student in developing a computational identity, the authors indicate that students must feel responsible for designing their own solutions, rather than working towards a single, predetermined correct answer. In terms of supporting students as they work towards digital empowerment, the authors encourage educators to find authentic and personally relevant contexts for the students to code within, and to ensure that these contexts have the potential to impact their lives and the lives of those in their communities.

Computational Literacy

Before Wing (2006), diSessa published "Changing Minds: Computers, Learning, and Literacy" (2000) in which he describes his grand vision of computers and coding in schools as computational literacy (CL). Unlike computer literacy, which may involve turning on a computer or using a keyboard or mouse for basic software operation, diSessa's CL involves "infrastructural" changes in schools and in society as it is used in diverse scientific, humanistic, and expressive forms: "a computational literacy will allow civilization to think and do things that will be new to us in the same way that the modern literate society would be almost incomprehensible to preliterate cultures." (p. 5).

In 2018, diSessa continued to explain this big picture view of CL, specifically in the context of science, technology, engineering, and mathematics (STEM) education: "I view computation as, potentially, providing a new, deep, and profoundly influential literacy—computational literacy—that will impact all STEM disciplines at their very core, but most especially in terms of learning" (diSessa, 2018, p. 4).

An important dimension of diSessa's CL, and specifically its connection to the subjects of mathematics and science, highlights the education argument for coding, and is sometimes communicated as "coding to learn", rather "than learning to code" (Popat & Starkey, 2019). When coding to learn, students program a computer in order to learn concepts and skills associated with the context of the program. Rather than a focus on the final artefact that results from the code (the running program), the educator's focus is on the concepts and skills developed as the students engage in the development of the artefact. In "Computer Coding in the K–8 Mathematics Curriculum?", Gadanidis et al. (2017a, b) highlight how the integration of coding in mathematics creates pedagogical opportunities such as (1) making abstraction tangible, (2) automating processes and making dynamic models, and (3) creating educational contexts that allow for differentiated instruction and student agency. The value of automating processes and making dynamic models is highlighted in work by Wilkerson (Wilkerson-Jerde et al., 2015; Wilkerson et al., 2018) and Gadanidis (Gadanidis et al., 2017a, b; Gadanidis et al., 2019), where students use or build computational models and simulations in order to better understand mathematical, scientific, and engineered systems. Wilkerson and Fenwick (2017) believe that CS shares language with mathematics that can be used to represent models using precise language resulting in a description of patterns and processes.

More recently, Kafai and Proctor (2022) developed three framings for CT, and concluded that in an effort to avoid "overloading the concepts of computational thinking with multiple meanings" (p. 148), they now adopt a framework of CT in their work that helps clarifies three main questions related to CS and K-12 education: (1) who should learn CS?, (2) what should be learned in CS?, and (3) how should learning occur in CS?

A Foundation for Analysis

Coding and computational thinking concepts and skills are impacting educational policy in Canada and around the world. In Canada, $110 million was allotted to the CanCode initiative which aims to engage over 2 million young people from K-12 in coding and digital skills development (Department of Finance Canada, 2019), while in the US the Computer Science for All initiative, which was first announced in 2016 by then President Barack Obama, is intended to empower American students from K-12 to learn CS (Smith, 2015). In 2016, the European Commission, Joint Research Centre published their report "Developing Computational Thinking in Compulsory Education: Implications for Policy and Practice" (Bocconi et al., 2016) that acknowledged the increased attention that CT and related concepts were receiving in education, and provided a comprehensive overview of CT

and related skills for the younger grades. In addition, CT is now included in the Organisation for Economic Co-operation and Development's Program for International Assessment (PISA).

Within the context of these initiatives, there is a need to ensure that coding and CT-related concepts and skills are integrated into various subjects using evidence-based approaches. In Zhang et al.'s "There is an Evidence Crisis in Science Educational Policy" (2022), the authors focus on "exploration based pedagogy" in science, a pedagogy that is similar to the "coding to learn" (Popat & Starkey, 2019) approach introduced earlier. Zhang et al. (2022) explain that "exploration based pedagogy", also referred to as inquiry, discovery, or problem-based learning, is often reflected in science education practice and policy, but not adequately supported by evidence.

> We should never use program-based studies as the sole source of evidence for any particular instructional procedure such as inquiry-based learning. All such recommendations should also include randomized controlled trials and large-scale correlational studies. However, program-based studies have been relied on almost exclusively in the standards to recommend inquiry-based learning with almost no questions addressed about the less favorable results from correlational and controlled studies. It is troubling to see sweeping curriculum reforms reinforced and overarching claims accepted while a large number of critical data sets have been ignored. (Zhang et al., 2022)

This issue is raised here to make clear the intentions of this article. This article is meant to inform the current understanding of curriculum development in Canada by considering the changes taking place as a result of the integration of coding and computational thinking in the K-8 grades. This article uses arguments and perspectives to analyse recent curriculum initiatives, and to further conversations surrounding the integration of coding and computational thinking in schools. It does not set out to provide empirical evidence related to the effective integration of coding and CT concepts and skills, instead it seeks to provide a better understanding of this current phenomenon.

Problem Description

Considering the theoretical approaches to coding in K-8 education discussed by leading researchers in the field, and considering the various goals and rationale for coding from jurisdictions outside of Canada, it is important to identify, and develop an understanding of, the components of coding curriculum in Canadian jurisdictions. Without an in-depth analysis of recent curriculum initiatives, educators, researchers, and policy makers will lack clarity terms of:

- The placement of coding-related concepts and skills in existing curricula;
- The goals and rationale of coding curricula; and the
- The theoretical perspectives underpinning the various curricula.

Recently, two studies have been conducted that explore CT in K-12, Canadian education. Hennessey et al. (2017) analysed Ontario elementary school curriculum, searching for CT-related terms described by Brennan and Resnick (2012), and concluded that "while CT terms appeared mostly in mathematics, and concepts and perspectives were more frequently cited than practices, related terms appeared across almost all disciplines and grades" (p. 79). Additionally, Gannon and Buteau (2018) provide an effective, initial description of the integration of CT in Canadian provinces and conclude that there is a wide variety of integration models being implemented in the various provinces. The authors also conclude that there are a number of provinces that have begun curriculum revisions, or that have begun supporting the development of programmes and resources related to CS.

Considering these findings, this paper provides further analysis of Canadian curriculum, with an emphasis on not only CT concepts and skills, but with an emphasis on all coding-related contexts. It also hopes to add to the works of Hennessey et al. (2017) and Gannon and Buteau (2018) by investigating the goals and rationale, as well as the supported arguments or orientations, for learning coding represented in the various curricula in Canada.

Purpose and Research Questions

The purpose of this research is to provide a comparative analysis of coding-related curricula in the K-8 grades from various provinces. In order to do so, the article will answer the following research questions:

1. Where are coding, CT, and computer science concepts and skills currently found in Canadian, K-8 provincial curricula?
2. What are the expressed goals and rationale for the inclusion of coding, CT and computer science concepts and skills within Canadian, K-8 provincial curricula?
3. What are the learning arguments or orientations reflected in the coding, CT, and computer science components in Canadian, K-8 provincial curricula?

By answering these questions, this research provides educators, policy makers, and researchers with an analysis of current coding, CT, and CS curriculum initiatives in the K-8 grades and will add an important Canadian perspective to existing international studies. It will also provide groundwork for potential, future curriculum development as well as foundational knowledge to help research and policy surrounding the implementation of this curricula.

Theoretical Frameworks and Methodology

This study will employ comparative document analysis implemented within the theoretical framework of constructivism that views learning as an interpretive and iterative process of building, done by active learners interacting with the world (Fosnot, 1996).

Constructivism

This research employs constructivism as its foundational theoretical framework, which involves epistemological beliefs whereby individuals develop subjective meanings of their experiences, resulting in knowledge being built, rather than found (Creswell & Creswell, 2013; Merriam & Tisdell, 2015). A constructivist approach considers knowledge as something that is constructed in the mind of the learner, and that "fits" with reality (Bodner, 1986). Constructivism is a popular worldview or approach to qualitative research, and includes the following assumptions, identified by Crotty (1998):

1. Human beings construct meanings as they engage with the world they are interpreting;
2. Humans engage with their world and make sense of it based on their historical and social perspectives, which has implications when one considers both those being researched (perhaps students, or educators), as well as the individual conducting the research themselves;
3. The basic generation of meaning is always social, arising in and out of interaction with a human community.

Constructivism is a popular framework for qualitative research, and one that is appropriate for this type of study considering the subjective nature of the document analysis. An alternative approach and research design might involve a more quantitative methodology employing a positivistic perspective. This might include the counting of coding categories as they develop, or some type of numerical weighting. Considering the small number of documents involved in this study, and the relative size of each, it is believed that the counting or weighting of categories, while providing objective and quantifiable data, would not provide better understanding or improved insights related to the curriculum documents in question.

Methodology and Document Analysis

In order to effectively answer the research questions in this study, the methodology employed involved an initial analysis of K-8 curriculum from all Canadian provinces, with the intention of identifying where coding concepts and skills have been included. Curriculum from all Canadian provinces was analysed, with a focus on identifying where coding or related concepts were included. All documents were retrieved online, from open-access government (e.g. Ministry of Education) websites. The curriculum documents outline the learning objectives, or expectations, that are to be met in each respective province. Curricula from Yukon, North West Territories, and Nunavut were not included in this analysis as they implement curricula from various provinces including British Columbia, Alberta, Saskatchewan, and Manitoba (Government of Yukon, 2022; Government of Northwest Territories, 2021; Nunavut Department of Education, 2019).

Once the initial list of documents was identified, a more in-depth analysis took place involving the identification and analysis of all explicitly stated goals and rationale of the curricula. Following this identification of curricula, and the analysis of stated goals and objectives, document analysis provided insight into the teaching and learning orientations of the various curricula.

Document analysis involves systematic procedures for reviewing and evaluating documents in order to elicit meaning, gain understanding, and develop empirical knowledge (Bowen, 2009; Corbin & Strauss, 2008). It is an iterative process that includes finding, selecting, appraising, and synthesizing data contained in documents, and is often combined with content and thematic analyses (Bowen, 2009). The content analysis aspect of the study involved preliminary coding, which is the organizing of information from the documents into categories related to the central questions of the research (Bowen, 2009). This included where explicit goals and rationale of the curriculum were identified, as well where learning outcomes or expectations were expressed.

In this study, the curriculum policy documents from Canadian provinces were analyzed, which are all organized in a similar fashion, with grade levels and specific subject areas identified. As stated, a preliminary scan of these documents, related to the K-8 grades, was conducted, identifying documents that include coding, CT, and CS concepts. These documents were then selected for content and thematic analyses, which involved a thorough and repeated analysis of the documents, the coding of categories, the redefinition and organization of these categories, and the development of emerging themes. The coding process and the development of themes were influenced by the theoretical approaches and arguments for coding presented earlier. Key terms from the literature served as guides for initial categories and specific wording from curricula was compared to the theoretical perspectives and the arguments and goals for coding in the younger grades (Table 1).

Findings

The findings for each of the provinces have been organized according to the placement of coding, CT, and CS concepts in the K-8 curricula, the explicitly stated goals and rationale, and the learning orientations. These findings are listed below, in Table 3, from West to East, as they would be presented on a map, and are then expanded upon in the next section.

Table 3 Location, goals, grades, implementation type, and learning orientations of coding related expectations in curricula

	Location	Goals	Grades	Implementation	Learning orientations
British Columbia	Applied design, skills, and technologies (ADST)	Practical and applied focus in the area of technology	6–8	Optional component	Computational thinking and robotics
Alberta	Science	Apply computational thinking in order to solve problems and perform scientific inquiries	K-6	Mandatory component	Computer science and computational thinking
Ontario	Mathematics	Develop algebraic reasoning and provide opportunities to apply and extend mathematical thinking, reasoning and communicating	1–8	Mandatory component	Solve problems and create computational representations of mathematical situations
	Science and technology	Explore science and technology concepts and learn specific skills	1–8	Mandatory component	Write and execute code in investigations and when modeling concepts
Quebec	Science and technology	Develop competencies and hands-on learning	5–6	Mandatory component	Robotics
New Brunswick	Technology	Practical skills in technology in order to prepare for life and the career choices required in a modern economy	6–8	Mandatory Component	Creating technologies (app development, games, programs)
Nova Scotia	Information and communication technology	Problem-solving and innovation	4–6	Mandatory component	Computer science and computational thinking
Newfoundland and Labrador	Technology education	Technological literacy	8	Mandatory component	Control technology, robotics and automated systems

British Columbia's Applied Design, Skills, and Technologies Curriculum

In British Columbia, coding-related concepts and skills are found in the applied design, skills, and technologies (ADST) grades 6–8 curriculum (British Columbia Ministry of Education, 2016a). While the ADST curriculum begins in grade 1, specific content for the 1–5 grades is not listed and instead, teachers are meant to draw content from other areas of learning, in a cross-curricular fashion. In grades 6–8, specific content is listed in the form of 12 different modules (some of which include coding-related concepts and skills). In grades 6–7, teachers select a minimum of three content modules from the list of 12. In grade 8, schools can select one, or several modules, to make up the equivalent of a full-year course in ADST.

The coding-related modules that may be selected include Computational Thinking and Robotics. Other modules, such as Computers and Communications Devices and Digital Literacy, while related to computers and technology, do not include concepts and skills specific to coding, CT, or CS. In grade 8, schools are meant to provide students with a full-year course in ADST that can be made up of one or more of the 12 modules. Schools also have the choice of developing their own modules that include locally developed content, and that can be used instead of, or in addition to, the modules provided.

The stated goals and rationale for British Columbia's Applied Design, Skills, and Technologies curriculum highlight a very practical and applied focus. The curriculum is meant to "foster the development of skills and knowledge to support students in developing practical, creative, and innovative responses to everyday needs and challenges" (British Columbia Ministry of Education, 2016b). The learning opportunities are designed to allow students to discover their interests in practical and purposeful experiences and are built upon the assumption that students have a desire to create and work in practical ways.

The CT modules indicate that students will be provided with the opportunity to learn visual programming in grades 6 and 7 (such as a block-based language like Scratch), as well as text-based programming in grade 8. In terms of the specific subject matter, the CT module includes learning and teaching related to algorithms, sequential instructions, programming, debugging, and the visual representations of problems and data, which all connect to the literature in terms of associated CT concepts or skills. Grades 6–7 subject matter also includes students using visual programming, which could be taught using the Scratch programming language, as it has been identified as an effective way to help students engage in CT activities (Zhang & Nouri, 2019).

Alberta's Science Curriculum

In the spring of 2021, the Alberta government released draft curriculum that included K-6 science expectations related to coding (Alberta, 2021). Two years later, the final curriculum was released that maintained the coding components from the draft (albeit slightly altered), with mandatory implementation of grades K-3, and optional implementation of grades 4–6, in September 2023 (Alberta Education, 2023).

Computational thinking is listed as one of the major changes to Alberta's K-6 Science curricula (Alberta Education, 2021). The draft curriculum website acknowledges that the old curricula did not have any references to problem solving with coding, whereas the new curricula include "clear expectations for students to learn problem solving that includes coding and algorithms" (Alberta Education, 2021). Computer science plays a prominent role in the curricula, as the document's overview includes the discipline alongside physics, chemistry, biology, Earth science, and astronomy. The overview reflects a desire for students to develop critical thinking and problem solving skills and encourages students to use their curiosity, creativity, and perseverance. The overview also acknowledges that studying science can enable students to evaluate information they encounter every day and can lead to careers in research, medicine, CS, geology, engineering, astronomy, agriculture, and more.

Computer science has a large footprint in the K-6 Science curricula. The content in the document is grouped into the following five main categories, with CS appearing alongside more traditional scientific areas of study.

- Matter
- Energy
- Earth systems
- Living systems; and
- Computer science

This makes it clear that the learning of CS concepts and skills is an important goal of this curriculum. As previously stated, critical thinking and problem solving skills appear to be important goals of the curriculum, and the learning surrounding CS in the document is focussed on these areas.

Each grade in the K-6 Alberta Science curricula includes a single guiding question and learning outcome for the category of CS. These are listed in Table 4. It is clear that the themes of instructions, creativity, design, and abstraction are key components of the learning outcomes. In Kindergarten and grade 1, students learn about following, creating, and the influence of *instructions*. In grade 2, students consider the use of creativity in instructions and in grade 3 they investigate the relationship between *creativity* and CT. In grades 4 and 5, the focus shifts to *design* in order to resolve problems and achieve specific outcomes or purposes. Finally, in grade 6, students consider the CT concept of *abstraction.*

Ontario's Mathematics and Science and Technology Curricula

Ontario is the only jurisdiction with studies that include CS-related concepts in both its Mathematics and Science and Technology curricula. In 2020, Ontario released new grades 1–8 Mathematics curriculum that is the first, and only, Ontario elementary curriculum document to include explicit coding-related concepts and skills (Ontario Ministry of Education, 2020). The curriculum document is divided into six distinct but related strands including Social-Emotional Learning (SEL) Skills in Mathematics and the Mathematical Processes, Number, Algebra, Data, Spatial Sense, and Financial Literacy. The curriculum includes both overall and specific curriculum expectations. The 13 overall expectations, which are common for each grade, "describe in general terms the knowledge, concepts, and skills that students are expected to demonstrate by the end of each grade". The specific expectations, which are different in each grade, "describe the expected knowledge, concepts, and skills in greater detail" (Ontario Ministry of Education, 2020, p. 18). The coding expectations are found in Strand C – Algebra, but it is important to note that the accompanying curriculum context document indicates that the coding expectations can be applied across all strands, and is meant to provide students with opportunities to apply and extend their math thinking, reasoning, and communicating (Ontario Ministry of Education, 2020).

The vision of the Ontario Mathematics 1–8 curriculum is to help students develop a positive identity as skilled mathematics learners, to support them as they use mathematics to make sense of the world, and to enable them to use mathematics to make sound decisions (Ontario Ministry of Education, 2020). Coding is mentioned as a means for students to develop algebraic reasoning, and also to provide students with opportunities to "apply and extend their math thinking, reasoning and communicating" (Ontario Ministry of Education, 2020, p. 34). This reflects Papert, diSessa, Wilkerson, and Gadanidis's view of coding or CT as being an important component in mathematics education, and as a tool that can allow students to not only solve mathematical problems, but also to experience and engage with mathematical concepts.

The coding expectations in Ontario's grades 1–8 Mathematics curriculum emphasize that students will be writing, executing, reading, and altering code, which hints at a very action-oriented type of

Table 4 Computer science guiding questions and learning outcomes in Alberta's K-6 Science curriculum

	Guiding question	Learning outcome
Kindergarten	How can instructions be used?	Students interpret instructions in various environments
Grade 1	How can instructions affect outcomes?	Students follow instructions and relate them to outcomes
Grade 2	How can creativity support design?	Students apply creativity when designing instructions to achieve a desired outcome
Grade 3	How does creativity contribute to computational thinking?	Students investigate creativity and its relationship to computational thinking
Grade 4	How can design meet needs?	Students examine and apply design processes to meet needs
Grade 5	In what ways can design be used to help achieve desired outcomes or purposes?	Students create and justify a design that could be used by a human or machine to address a challenge
Grade 6	In what ways are abstraction, design, and coding related?	Students create and refine computational artefacts through the use of design and abstraction

learning, where students can potentially learn mathematics by coding. The overall curriculum expectation, which spans grades 1–8, involves using coding concepts and skills to solve problems and create computational representations of mathematical situations. This expectation is interesting as it does not indicate what types of mathematical situations are meant to be solved or created. Considering that coding is meant to be applied across various strands, as indicated in the curriculum context, one is to assume that the mathematical context for these problems and representations can be drawn from the rest of the curriculum.

In addition to the overall expectations, each grade from 1 to 8 includes two specific expectations related to coding that involve students writing code, as well as reading and altering code. This emphasis on reading, altering, writing, and executing code is similar to the pattern of engagement for novice computer programmers called Use-Modify-Create, which was first described by Lee et al. (2011) in "Computational Thinking for Youth in Practice".

Within Ontario's new Science and Technology curriculum (2022), coding concepts appear in all grades, from 1 to 8, within a broad Strand A (STEM Skills and Connections) that sits atop the remaining four strands (Life Systems, Matter and Energy, Structures and Mechanisms, and Earth and Space Systems). The Strand A components are described as "foundational STEM skills and connections that will enable students to investigate concepts and integrate knowledge from each of the other strands and to make practical connections between science and technology and other subject areas" (Ontario Ministry of Education, 2022, p. 66). As educators integrate the coding concepts across the various strands, they are encouraged to support students in using coding in investigations and to model science and technology concepts. The curriculum document explains that this integration provides students with a hands-on, experiential way to (1) learn about concepts, (2) do science, (3) develop solutions to problems, (4) demonstrate their learning, (5) learn about the digital world around them, (6) take pride in their work, (7) provide an opportunity for agency in their learning, and (8) realize that they can shape the future in positive ways (Ontario Ministry of Education, 2022). Interestingly, these are all affordances that are discussed in the literature related to coding, CT, and CS.

The specifics of the learning outcomes in Ontario's grades 1–8 Science and Technology curriculum include students writing and executing code in investigations and when modeling concepts, as well as identifying and describing the impacts of coding and of emerging technologies. Like in Ontario's Mathematics curriculum, each grade provides a specific coding or CS-related concept to be explored. These include, from grades 1 to 8, (1) creating clear and precise instructions; (2) decomposing problems into smaller steps; (3) testing, debugging, and refining programs; (4) producing different types of output; (5) using different methods to store and process data; (6) obtaining input in different ways; (7) planning and designing programs; and (8) automating large systems in action.

Quebec's Science and Technology Curriculum

In Quebec, the only coding-related curriculum in the K-8 grades appears in the elementary Science and Technology curriculum where there is essential knowledge related to students recognizing robotic structures that use servomechanisms (grades 5 and 6), as well as recognizing the impact of electric appliances, where microprocessors and computers are listed in brackets as examples (grades 3, 4, 5, and 6) (Québec Ministère de l'Éducation, 2009).

With very little coding-related curriculum concepts in the K-8 grades, the Quebec curriculum does not explicitly state any aims or goals related to the use of coding. The main Quebec Education Program document does state, however, that two characteristics of the Quebec Education Program are the development of competencies and recognizing that learning is an active process (Québec Ministère de l'Éducation, 2001).

The curriculum components related to students recognizing robotic structures that use servomechanisms (grades 5 and 6) and impact of electric appliances, where microprocessors and computers are listed in brackets as examples could allow for teachers to include coding concepts and skills in their instruction; however, it is also possible for this not to occur and still have students meet the requirements of the curricula. This is surprising considering the stated characteristic of the Quebec Education Program being the development of competencies and recognizing that learning is an active process.

New Brunswick's Technology Curriculum

The New Brunswick elementary curriculum includes a 2016 pilot document where coding plays a predominant role. In Middle School Technology Education, coding is listed as one of three main subject areas for grades 6–8 technology instruction, alongside Computer operations and Projects work (New Brunswick Department of Education and Early Childhood Development, 2016). The Conceptual Framework Divisions section of the document lists a number of digital technology skills for students to learn (including file management, coding/programming, computer aided drafting, video and audio production, and digital citizenship), and indicates that coding should take up a minimum of 10% of each of the grades 6, 7, and 8 years.

The General Curriculum Outcomes (GCOs) and Specific Curriculum Outcomes (SCOs) span across the three grades (6, 7, and 8) and include three main areas: (1) technological operations and concepts; (2) critical thinking and problem-solving skills; and (3) responsible citizenship. The second main area, critical thinking and problem-solving skills, is where coding and related concepts and content are found. This section, which again, is meant to span across grades 6, 7, and 8, includes the following two specific outcomes: "2.2 Students will examine data to draw conclusions and recommend solutions to improve performance" (New Brunswick Department of Education and Early Childhood Development, 2016, p. 15); and "2.5 Students will understand and demonstrate computer coding/programming concepts and terminology" (New Brunswick Department of Education and Early Childhood Development, 2016, p. 15). Coding is listed in the concepts and content section of SCO 2.2, while app development, robotics, game development, and electronics are all listed in the concept and content section for SCO 2.5.

The Middle School Technology Education document reflects the economic argument for coding as it indicates that grade 6 to 8 students require a wide variety of practical skills in technology in order to prepare for life and the career choices required in a modern economy. The document indicates that the coding area of study, often seen as "the mysterious side of technology usage" (New Brunswick Department of Education and Early Childhood Development, 2016, p. 4), is recognized as strengthening logical thinking and problem solving skills, which connect to CT concepts, even though CT concepts and practices are not mentioned further in the document.

New Brunswick's specific outcomes related to coding include using code to examine data and draw conclusions, and having students "understand and demonstrate computer coding/programming concepts and terminology" (New Brunswick Department of Education and Early Childhood Development, 2016, p. 15). The terminology used in the outcomes indicates that students will be both actively programming a computer or physical digital device, as well as demonstrating knowledge surrounding related terminology. In addition, app development, robotics, game development, and electronics are all mentioned as concepts and content, which ensures that students will be focused on actively creating projects or artefacts with code. The connection of coding to data would potentially require a cross-curricular approach, in which mathematics concepts appropriate to the grade may be used, in order to draw relevant conclusions.

Nova Scotia's Information and Communication Technology Curriculum

In Nova Scotia, the province currently has two Information and Communication Technology curriculum documents, one for primary to grade 3 (P-3) and one for grades 4 to 6. The P-3 document lists essential learning outcomes and performance indicators related to digital citizenship and productivity, but coding-related concepts and skills are never explicitly mentioned. In the grade 4–6 document coding is listed as an explicit outcome, where students will understand and apply the basic concepts of CS, including algorithms, abstraction, and computational thinking (Nova Scotia Department of Education and Early Childhood Development, 2016a).

In Nova Scotia's 2016 Action Plan for Education Annual Report, coding was acknowledged as promoting skills such as problem-solving and innovation, which were both linked to growth industries like "computer programming, marine industries, and manufacturing" (Nova Scotia Department of Education and Early Childhood Development, 2016b, p. 4). In the 4–6 grades, the coding outcomes better reflect the economic and educational goals, as robotics controls, gaming, problem solving, communication, and specific computer programming concepts are all listed as grade-specific strategies and skills.

Students in grades P-3 may code a computer in class; however, the curriculum does not explicitly make this a mandatory proposition. The curriculum documents make reference to the safe operation of computer and digital devices; however, this could just as easily include digital presentation or spreadsheet software, or even effectively carrying out internet searches. In the 4–6 grades, the learning orientations related to coding are clear, as in each grade, students will "understand and apply the basic concepts of CS, including algorithms, abstraction, and computational thinking" (Nova Scotia Department of Education and Early Childhood Development, 2016a). This outcome highlights the need for students to understand specific CS and programming concepts (such as conditional statements, loops, variables, and programming languages), as well as CT concepts (such as pattern recognition, sequencing, debugging, efficiency, and abstraction). In addition, the control of robotics, gaming, and real-world situations are also highlighted, allowing for a variety of contexts where students can learn and apply the CS and CT concepts (Nova Scotia Department of Education and Early Childhood Development, 2016a).

Newfoundland and Labrador Technology Curriculum

Within the area of technology education, Newfoundland and Labrador K-8 curriculum includes a grade 7 Communications Technology Module that makes reference to students identifying examples of technologies encoding and decoding information (Newfoundland and Labrador Department of Education, 2002); however, coding in terms of programming a computer is not explicitly mentioned. In grade 8, a Control Technology Module exists that includes coding-related concepts and skills (Newfoundland and Labrador Department of Education, 2006). Students must complete the grade 7 Communications Technology Module and a Grade 8 Production Module before progressing to the grade 8 Control Technology Modules.

As indicated in the front matter of the curriculum Control Technology Document, the focus of the curriculum is the development of student's technological literacy, capability, and responsibility: "Students will be exposed to many facets of technology and will gain literacy through active participation in knowledge acquiring and skill developing activities presented throughout the implementation of the Grade 8 Control Technology Module" (Newfoundland and Labrador Department of Education, 2006). The active process of learning is emphasized throughout the document, as is a focus on coding being used as a practical skill to control systems and devices.

The curriculum outcomes themselves are written in a way that may lead to students discussing programming rather than actually programming a computer (ex: 1.17 define programming in terms

of communications within control technology systems, 1.18 describe the function of specific simple programs). The document, however, provided added information for teachers, in terms of organization and presentation, which includes the following explanation:

> [p]rogramming in the Grade 8 Control Technology Module is of an introductory nature and is meant to provide students with a basic communications system that can enable them to construct functional control technology systems. Students need to understand that programming is a means of developing a set of operations that specify what a particular mechanism or system should accomplish. (Newfoundland and Labrador Department of Education, 2006)

This description confirms that students will be programming a computer within the context of a controls or robotics system; however, it is one of the only references whereby it is clearly stated that students will code, rather than simply discuss or identify code components and applications.

Comparative Analysis and Discussion

Coding or Coding-Related? For Some or For All?

After analyzing the location and type of implementation of coding expectations in K-8 curricula from the various provinces in Canada, it is apparent that four main categories are represented. These are expressed in Table 5. A fifth category, number 2, has been added in Table 5, and while there are no provinces that make up this category, it has been added as a possible category that fits within this framework. This category emerged during the analysis process, as it was realized that a jurisdiction could develop expectations that combine components from both category 3 and category 4. In category 3, provinces such as Quebec and Newfoundland and Labrador include expectations that could lead to students potentially programming a computer, but do not explicitly state this as an outcome. In category 4, British Columbia includes optional coding expectations that may or may not be experienced by students, depending on which "modules" are selected to be taught. Category 2 emerged by considering a jurisdiction that developed expectations that hint at, but do not explicitly state that students will program a computer (similar to Quebec and Newfoundland and Labrador in category 3), and that include these expectations in an optional module (similar British Columbia in category 4).

Category 2, therefore, includes jurisdictions where curriculum expectations might be found in an optional component or module, and where the expectations are written in such a way that could allow for

Table 5 Categories of implementation of coding expectations in Canadian K-8 curricula

1. Jurisdictions that do not include any coding-related expectations	• Saskatchewan • Manitoba • Prince Edward Island
2. Jurisdictions that include coding-related expectations that could potentially lead to coding experiences for some students	None identified
3. Jurisdictions that include coding-related expectations that could potentially lead to coding experiences for all students	• Quebec • Newfoundland and Labrador
4. Jurisdictions that include coding-related expectations that guarantee coding experiences for some students	• British Columbia
5. Jurisdictions that include coding-related expectations that guarantee coding experiences for all students	• Alberta • Ontario • New Brunswick • Nova Scotia

a teacher or student to program a computer, but this may not be explicitly stated. An example might be a jurisdiction that includes expectations surrounding an awareness of how computer algorithms work, and then includes this expectation in a module that is not mandatory across the jurisdiction. Some students may be offered this module, but not all, and some students who are offered this module might program a computer to learn about this concept, but it is possible that they do not.

Category 3 is similar to category 2, in that the curriculum expectations could allow for a teacher or student to program a computer, but this may not be explicitly stated. The difference between category 2 and category 3 is that in category 3 all students will experience the curriculum expectations, as they are part of mandatory learning for all students.

Category 4 includes jurisdictions where the expectations or outcomes are written in a way that guarantees that students will be programming a computer, but the expectation appears in an optional component of the curriculum. An example of this might be British Columbia's Computational Thinking module that appears in the ADST curriculum. This module is one of 13 optional modules, so not all schools or teachers will select the module, but once selected, the module includes students learning visual programming, which explicitly states that students will program a computer.

Finally, category 5 involves curriculum expectations that are written in a way that ensure that students will program a computer in order to meet the expectations, and they are found in part of the curriculum that is taught to all students. An example of this would be the expectations found in Ontario's Mathematics and Science and Technology curricula and the expectations in Alberta's Science curriculum. These curricula are mandatory for all students to learn, and the wording clearly indicates that students will be required to program a computer in order to meet the expectations (Table 5).

The reason for the importance of these categories is for policy makers to understand the impact of potential coding curriculum, and to consider implementation. This paper began by presenting Webb et al. (2017), Passey (2017), Vogel et al. (2017), and Hubweiser et al.'s (2015) arguments for coding in the younger grades, but if one is to believe that these arguments are valid and important for all, then implementation should represent category 5 of Table 6, where coding-related expectations guarantee coding experiences for all students. Developing coding expectations that may or may not be experienced by all students or developing coding-related expectations that may or may not lead to students experiencing the power of programming a computer would not suffice. Likewise, the theoretical approaches presented at the beginning of the paper make it clear that the coding concepts and skills have value for all students, whether from a computational thinking, fluency, participation, literacy, or action perspective, which is why the classification of category 5 is so important, as it ensures that all students in a jurisdiction will experience programming a computer.

If a goal for a policy maker is for students to program a computer, then the expectations and outcomes should be written in clear language that signals to educators the students will program a computer, rather than discuss programming a computer. Likewise, if the goal is for all students to be provided with the opportunity to program a computer, then policymakers need to ensure that expectations and outcomes are placed in curriculum documents that include mandatory learning, rather than optional modules or courses. If modules or courses are optional, then it is possible that a number of students miss out on the opportunity to be exposed to coding concepts and skills.

Another way to consider categories 2, 3, 4, and 5 is presented in Fig. 1.

Coding on Its Own or Integrated… Somewhere?

Document analysis reveals that coding expectations in the K-8 curriculum from Canadian provinces appear to be integrated in four different ways:

Table 6 Theoretical perspectives reflected in provincial coding-related curricula

Theoretical perspectives	Curriculum
Constructionism (Harel & Papert, 1991; Papert, 1993) • Building knowledge structures, like constructivism, but doing so through the "construction" of a public entity • Using objects to think with • Recognizing the computer as the "Proteus of machines" to support the culture of the classroom that may be missing	**BC:** • Applied design is at the heart of the BC curriculum, with CT being implemented within the context of an experiential, hands-on program of learning through design and creation • Curriculum rationale states that the ADST curriculum harnesses the power of learning by doing • Introduction states that applied learning is part of all of the ADST curricula, through the Curricular Competencies that make up the "doing" part of the curricula **Alberta:** • A central, organizing idea of curriculum is that problem solving and scientific inquiry are developed through the knowledgeable application of creativity, design, and computational thinking **Ontario:** • Technology is recognized as having changed how students can interact with mathematics and science and technology concepts • Coding provides students with the opportunity to apply and extend math thinking, reasoning and communicating, and to investigate and model science and technology concepts
Computational Thinking (Wing, 2006; Grover & Pea, 2018) • Solving problems using concepts and strategies related to CS • Includes CT concepts such as logical thinking, algorithms, patterns, abstraction, evaluation, and automation • Includes practices such as decomposing a problem, creating computational artefacts, testing and debugging, iteration, collaboration, and creativity	**BC:** • Module title is Computational Thinking • Simple algorithms that reflect CT (grades 6–7) • Visual representations of problems and data (grades 6–7) • Debugging algorithms and programs by breaking problems down into a series of sub-problems (grade 8) **Alberta:** • The components and importance of instructions are analyzed in early grades (K-3) • Computational thinking components and the term itself are included in grade 3 • Concept of abstraction is included in grade 6 and applied within the design context **Ontario:** • Concepts such as sequencing, concurrent events, repetition, testing and debugging, conditional statements and efficiency reflect components of the CT concepts • Students read and alter code and predict potential outcomes which reflect testing, debugging, and iteration included in the CT practices **New Brunswick:** • Coding recognized as strengthening logical thinking and problem solving skills **Nova Scotia:** • The learning outcome for grades 4–6 includes understanding and applying the basic concepts of CS, including algorithms, abstraction, and computational thinking • Performance and assessment indicators related to the outcome include organizing a sequence of events, debugging, and predicting outcomes

Table 6 (continued)

Theoretical perspectives	Curriculum
Computational Fluency (Resnick, 2018) • Includes student creativity and expression with digital tools • Students develop a voice and an identity through coding • Digital technologies are a symbol of possibility and progress and as students design and code they see themselves as part of the future	**BC** • Curriculum goals include students developing a sense of efficacy and personal agency about their ability to participate as inventors and innovators, reflecting social advantages of learning to code **Alberta** • Creativity serves as a major component of the curriculum; however, this creativity is in the context of problem solving rather than in the form of personal expression, or the social advantages of developing personal voice and identity
Computational Participation (Kafai, 2016) • Includes a focus on coding as a social practice • Includes collaboration, sharing of projects and the development of communities • Moves from building code to creating sharable applications	**Alberta** • In grade 5 students learn about and engage in collaborative processes in CS and the value of sharing ideas for effective design
Computational Action (Tissenbaum et al., 2019) • An alternative to a fundamentals approach, that instead focusses on project connecting to student's lives • Focussed on key dimensions of student identity and empowerment • Strives for the development of a critical consciousness as students create projects for their communities	**BC** • Curriculum goals include students becoming agents of change able to address practical challenges in a rapidly changing world
Computational Literacy (diSessa, 2018) • A big picture view of a change in STEM education (especially mathematics and science) with a new form of literacy • Literacy means that a representational form for supporting intellectual activities is adopted by a broad cultural group	**Ontario** • Curriculum documents indicate that coding can be incorporated across all strands and provides students with opportunities to apply and extend their math thinking, reasoning, and communicating, as well as investigating and modeling science and technology concepts • Curriculum documents indicate that as students progress through the grades, their coding experiences also progress, from representing movements on a grid, to solving problems involving optimization, to manipulating models to find which one best fits the data they are working with in order to make predictions • The overall expectations include solving problems and creating computational representations of mathematical situations using coding concepts and skills • The specific expectations include a progression of coding concepts such as repetition, conditional statements, and subprograms • The coding expectations take on the representational form, the associated learning in the grade takes on the intellectual activities, and the broad cultural group are the Ontario students and educators themselves

1. A component in technology curriculum (British Columbia, Ontario, Quebec, New Brunswick, and Newfoundland and Labrador)
2. A component in Information and Communications Technology curriculum (Nova Scotia)
3. A component in Science curriculum (Alberta, Ontario)
4. A component in Mathematics curriculum (Ontario)

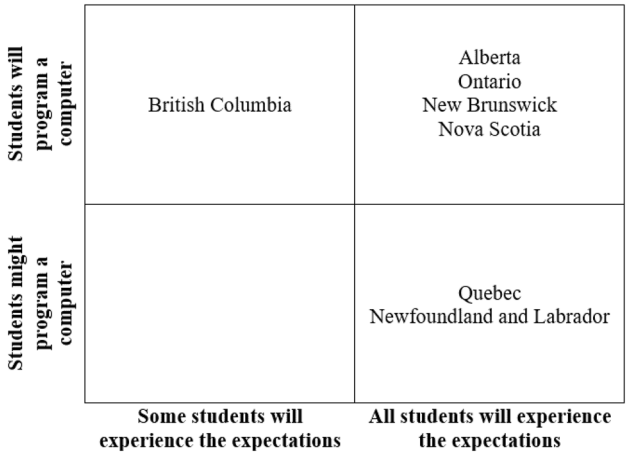

Fig. 1 K-8 coding curriculum implementation examples from Canadian provinces

While there perhaps is not a "correct" location to place coding-related concepts and skills in K-8 curriculum, what has become clear in this study is that the placement (and wording) of the expectations and outcomes should honour the subject area in which they are placed, as well as the stated goals. This point can be illustrated by comparing British Columbia's Computational Thinking module from the Applied Design, Skills, and Technologies (ADST) curriculum to Ontario's coding expectations in the Algebra Strand of mathematics.

A major goal of the ADST curriculum involves supporting students as they develop practical, creative, and innovative responses to everyday needs and challenges (British Columbia Ministry of Education, 2016a), yet the computational thinking components of the curriculum include the evolution of programming languages, as well as the study of binary number systems. While these may be appropriate concepts for students to learn, they do not speak to the applied nature of the curriculum, and they may prove difficult in providing context for the applied design stages of the curriculum competencies. In contrast, the coding expectations within the Algebra strand of the Ontario Mathematics curriculum demonstrate clearly that students are coding within the context of the specific subject, by solving problems and creating computational representations of mathematical situations (Ontario Ministry of Education, 2020). This wording, and the specific concepts involved in each grade, also connect to the goals of the curriculum that include providing students with the skills to "think critically and creatively and see connections to other disciplines beyond mathematics, such as other STEM disciplines" (Ontario Ministry of Education, 2020).

Another example that speaks to the need to honour the subject area in which the coding expectations are placed is Alberta's science curriculum. Weintrop et al. (2016) have presented a framework for the integration of CT that includes the science classroom, and Gravel and Wilkerson (2017) have presented a specific example of grade 5 students using computational artefacts to explore physics concepts. Both these approaches recognize the value of computational artefacts to learn about and explore science concepts, yet interestingly the Alberta grade K-6 curriculum does not capture this affordance within its CS components. A major organizing idea of the curriculum is "Problem solving and scientific inquiry are developed through the knowledgeable application of creativity, design and computational thinking" (Alberta Education, 2023, p. 14), yet the examples do not connect the development of computational artefacts to science concepts and skills. While students learn about CS in terms of instructions, creativity, design, and abstractions, the learning outcomes and examples do not connect to science concepts that are included in other areas of the curriculum. This is a missed opportunity as the design and coding of computational artefacts present a valuable opportunity to learn science concepts (Sengupta et al., 2013).

In addition to honouring the subject area in which the coding expectations are placed, as well as the stated goals of the curriculum, coding expectations and outcomes should clearly reflect well-defined arguments for the inclusion of coding in the younger grades. If policy makers embody the economic argument for coding, then it follows that coding expectations and outcomes be placed in the curriculum in a manner that connects coding to potential careers, such as within technology curriculum documents. If, on the other hand, policy makers embody the educational, "coding to learn" argument then expectations and outcomes should be written in a way that allow other components of the curriculum (whether it be mathematics or science) to provide the context for students to be programming on a computer. Interestingly, the manner in which the CT modules was placed in BC's ADST curriculum introduces the idea that coding expectations and outcomes might have a value in supporting the stages of a design process. This connection of coding to the design processes has not been discussed extensively in literature, especially within the K-8 grades.

Connecting Theory and Curricula

This article began with a description of theory in the field of K-12 CS-related education exploring Papert's foundational learning theory of constructionism, as well as the various perspectives of computational thinking, fluency, participation, action, and literacy. While answering the indicated research questions laid out, the document analysis process also provided insight into how these differing approaches were reflected in the K-8 coding curriculum of Canadian provinces. Table 6 lists, and briefly describes, the theoretical perspectives introduced in this article, as well as the components of the various coding curricula from Canadian provinces that reflect these approaches. Grover and Pea's (2018) CT was used in combination with Wing's (2006), as Grover and Pea provide additional depth that Wing's CT was lacking. Components of the Quebec and Newfoundland and Labrador curricula that relate to coding were not included in Table 6 as these components were very technical in nature, relating specifically to robotics and controls, and these components were not explicit in having students program a computer.

Analysing these curricula through the theoretical lenses indicates that:

- The theoretical approach of CT is reflected in five major coding curricula in Canadian provinces, with BC, Alberta, and Nova Scotia using this term explicitly;
- Computational fluency, participation, and action are not significantly reflected in the coding curricula of Canadian provinces;
- Alberta curriculum is primarily CT focused, but there are small components in grades 5 and 6 that reflect computational fluency, participation, and action; and
- While Ontario curriculum reflects some CT components, the coding expectations and description in the curriculum context reflect a computational literacy perspective. It is evident that students are learning to code within the context of mathematics and science and technology, and that the coding concepts in the expectations of each grade serve the role of the representational form that diSessa (2018) states is required for a literacy.

Computational thinking is reflected in BC, Alberta, and Nova Scotia, with all three jurisdictions using the term and providing related expectations, outcomes, or references to specific concepts and skills. In Ontario, the mathematic coding expectations refer to computational thinking–related concepts including sequential, concurrent, repeating, conditional, and nested events, however; their use seems to reflect a CS-focused approach, rather than one that embodies computational thinking specifically. In addition, the term computational thinking is not used in the Ontario 1–8 mathematics curriculum document. What is reflected in Ontario's curriculum, however, is diSessa's computational literacy whereby coding is integrated into school subjects in much the same way that algebra has become a tool in science,

mathematics, and other subjects. Alberta's coding outcomes in the K-6 Science document emphasizes a CS- and CT-focused approach, but does not explicitly leverage the development of computational artefacts to learn related science concepts. The Alberta curriculum does, however, reflect computational fluency, participation, and action, albeit with a small footprint and not as explicitly as CT.

In British Columbia ADST curriculum's reflects an emphasis on "constructionism", and the design and creation of an artefact can provide educators with a valuable opportunity to promote computational fluency, participation, and action in their pedagogy. Likewise, British Columbia's inclusion of "uses of robotics in local contexts" within the robotics module provides educators with valuable opportunities to connect coding to the lives and communities of students.

Nova Scotia's ICT curriculum clearly outlines the purpose of the coding outcome as connecting to real-world situations which, like British Columbia, could provide educators with an opportunity to have their pedagogy and selected projects reflect computational fluency, participation, and action. In New Brunswick, the Middle School Technology Curriculum emphasizes project-based learning that includes real-world connections and that is student driven. Like in British Columbia and Nova Scotia, this allows educators to select pedagogy and projects that could embody the creativity, collaboration, sharing, and social change that is reflected in computational fluency, participation, and action.

As previously mentioned, in Quebec and Newfoundland and Labrador, the coding curriculum expectations and outcomes are situated within a robotics context and are more technically focused. This is not to say, however, that a creative and motivated educator could not have the robotics projects reflect the computational fluency, participation, and action approaches.

Broadening Coding and Computational Thinking, Beyond Optional, Secondary Courses

This article provides evidence of how the implementation of coding and computational thinking being broadened into the K-8 grades and into subject areas such as mathematics, science, and technology. Considering this phenomenon, it is important to ask what the impact of this expansion will be on secondary (grades 9–12) courses. Traditionally, coding and computational thinking concepts and skills were included in optional computer science (CS) courses at the secondary level (Floyd, 2022).

As more students are exposed to coding and computational thinking concepts and skills in the younger grades, will they be motivated to enroll in CS-related courses at the secondary level, as their interests have been piqued, or as they have gained confidence through early exposure to concepts and skills? Is it possible that this increased interest and confidence leads to increased enrolment in secondary CS courses? Or, having experienced CS concepts and skills in the K-8 grades, will students and parents feel as though foundational CS concepts and skills have already been integrated enough into other subject areas, and therefore there is no need to enroll in specialized CS courses? Extending these questions further, if CS concepts and skills have such applicability in other subject areas, is it possible that the integration of CS into other subjects leads to the demise of specialized, secondary CS courses? At the very least, the changes taking place in K-8 curricula point towards a need to now carefully consider the goals of, and rationale for, CS-specialized courses in secondary schools, as well as the concepts and skills being taught in these courses. Beginning with the theoretical perspectives, some may consider having the secondary CS courses reflect a more CS-centric approach, embodying Wing's (2006) computational thinking perspective or embodying an economic argument for CS education, that prepares secondary students for post-secondary programmes related to CS, as well as related jobs in the field. Unfortunately, this could possibly leave out important social, cultural, and personal connections that may not be able to be adequately explored if students only learn CS concepts and skills in other subject areas.

In terms of specific concepts and skills, Ontario provides a good example of how secondary CS courses will need to be altered to reflect changes in elementary curricula. The Ontario Mathematics curriculum includes control structures in grades 1, 2, and 3 (such as the sequencing and repetition of

instructions), as well as conditional statements (decisions) in grade 4. These concepts were included in all grade 10 courses in Ontario over the last 55 years (Floyd, 2022). As an example, the 1983 document includes an expectation that students will "write simple routines that will illustrate the three basic operations involved in the processing of information—sequencing, selection, and repetition" (Ontario Ministry of Education, 1983, p. 16), while the current grade 10 Computer Studies course in Ontario includes expectations where students "write programs that includes a decision structure for two or more choices" and "write programs that use looping structures effectively" (Ontario Ministry of Education, 2008, p. 36). How will curriculum expectations such as these, in introductory, secondary CS courses, need to be altered if, referring to the findings from this article, all students in Ontario are now writing, executing, reading, and altering code that includes sequential, concurrent, and repeating events, and conditional statements in grades 1, 2, 3, and 4 respectively (Ontario Ministry of Education, 2020)?

The broadening of CS concepts and skills into other K-8 subject areas and grades presents an exciting opportunity for a greater number of students to be exposed to CS, but this will inevitably lead to changes needed in the traditional delivery model of the secondary, optional CS courses. Researchers and policy makers involved in secondary CS education are well-advised to play close attention to curriculum changes in the K-8 grades and to carefully consider the potentially changing underlying goals and rationale for optional, secondary CS courses, and the concept and skills taught within these courses as students arrive to these courses with greater CS experience than in the past.

Conclusion

This article set out to determine the location of coding-related concepts and skills in Canadian, K-8 provincial curricula, as well as the goals and learning orientations of the expectations and outcomes. By doing so, it provides insight into how a variety of jurisdictions reflect various theoretical perspectives and approaches to coding and computer science education.

The document analysis reveals that coding expectations appear in Canadian, K-8 curriculum in four ways: as a component in technology curriculum, as a component in ICT curriculum, as a component in science curriculum, and as a component in mathematics curriculum. In terms of the specifics of the implementation, five main categories appear that range from jurisdictions with no expectations and outcomes, to those with expectations or outcomes that guarantee coding experiences for all students. In between these two extremes are categories that include expectations and outcomes that could potentially lead to students programming a computer, and expectations and outcomes that were optional and would have to be selected by a board, school, or teacher. In terms of the goals of the coding curriculum, it is clear that the economic and learning arguments for coding are most reflected in the curriculum from the various provinces, with only some referring to the social advantages of learning to program a computer. Learning orientations were focused primarily on computational thinking concepts as these are explicitly mentioned in three provinces, while computational fluency, computational participation and computational action are not explicitly mentioned, but can provide valuable context for pedagogy and projects within several jurisdictions. Computational literacy is reflected in one jurisdiction, as coding appears explicitly in K-8 mathematics curriculum not with the infrastructural change that diSessa said was required, but perhaps signaling a trend in this direction.

In terms of theory within the field, this article provides evidence of how theoretical approaches related to coding, CT, and CS education are being integrated into learning outcomes for students in various ways. A good example is the front matter of Ontario's Science and Technology curriculum that communicates to educators a number of affordances for coding in the science and technology contexts. The inclusion of these affordances can be seen as theoretical perspectives influencing the development of programmatic curriculum, and influencing the implementation of the coding expectations in classrooms.

In addition, the article provides a number of examples of how constructionism is, or is not, reflected in learning outcomes developed by these jurisdictions. While some require students to be actively programming a computer, others include expectations where students may analyze the impact of coding, or could potentially explore coding and computational concepts in an unplugged fashion, that may not embody constructionist ideals.

Together, these findings present a clear picture of the current landscape of coding-related concepts and skills in K-8 curriculum of Canadian jurisdictions, providing a foundational understanding of the organization, goals, and orientations of curricula, as well as reflected theories, upon which to further study the novel and popular phenomenon of broadening exposure to CS-related concepts and skills.

Declarations

Conflict of Interest The authors declare no competing interests.

References

Alberta Education. (2021). *Draft Science Kindergarten to Grade 6 Curriculum*. https://cdn.learnalberta.ca/Resources/content/cda/draftPDF/media/Science/Science-GrK-6-EN.pdf

Alberta Education. (2023). Alberta's K-6 Curriculum: Science. https://curriculum.learnalberta.ca/curriculum/en/s/sci

Ames, M.G. (2018). Hackers, computers, and cooperation: A critical history of logo and constructionist learning. *Proceedings of the ACM on Human-Computer Interaction* 2(CSCW), 1–19. https://doi.org/10.1145/3274287

Bocconi, S., Chioccariello, A., Dettori, G., Ferrari, A., Engelhardt, K., Kampylis, P., & Punie, Y. (2016). Developing computational thinking in compulsory education. European Commission, JRC Science for Policy Report, 68. https://komenskypost.nl/wp-content/uploads/2017/01/jrc104188_computhinkreport.pdf

Bodner, G. M. (1986). Constructivism: A theory of knowledge. *Journal of Chemical Education*, *63*(10), 873-878. https://doi.org/10.1021/ed063p873

Bowen, G. (2009). Document analysis as a qualitative research method. *Qualitative Research Journal, 9*(2), 27-40. https://doi.org/10.3316/QRJ0902027

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 Annual Meeting of the American Educational Research Association*, Vancouver, Canada.

British Columbia Ministry of Education. (2016a). *Applied Design, Skills and Technologies*. https://curriculum.gov.bc.ca/sites/curriculum.gov.bc.ca/files/curriculum/adst/en_adst_k-9_elab.pdf

British Columbia Ministry of Education. (2016b). *Applied Design, Skills, and Technologies* - Goals and Rationale. https://curriculum.gov.bc.ca/curriculum/adst/goals-and-rationale

Corbin, J., & Strauss, A. (2008). *Basics of qualitative research: Techniques and procedures for developing grounded theory* (3rd ed.). Sage.

Creswell, J. W., & Creswell, J. D. (2013). *Research design: Qualitative, quantitative, and mixed methods approaches*. Sage publications.

Crotty, M. (1998). *The foundations of social research: Meaning and perspective in the research process*. Sage Publications

Dagienė, V., Jevsikova, T., & Stupurienė, G. (2019). Introducing informatics in primary education: curriculum and teachers' perspectives. In *Informatics in Schools. New Ideas in School Informatics: 12th International Conference on Informatics in Schools: Situation, Evolution, and Perspectives, ISSEP 2019, Larnaca, Cyprus, November 18–20, 2019, Proceedings 12* (pp. 83–94). Springer International Publishing. https://doi.org/10.1007/978-3-030-33759-9_7

Denning, P. J. (2017). Remaining trouble spots with computational thinking. *Communications of the ACM, 60*(6), 33-39. https://doi.org/10.1145/2998438

Department of Finance Canada. (2019). *Investing in the middle class: Budget 2019*. https://www.budget.gc.ca/2019/docs/download-telecharger/index-en.html

diSessa, A. (2000). *Changing minds*. MIT Press.

diSessa, A. (2018). Computational literacy and "The Big Picture" concerning computers. *Mathematics Education, Mathematical Thinking and Learning, 20*(1), 3-31. https://doi.org/10.1080/10986065.2018.1403544

Floyd, S. (2022). *The Past, Present, and Future Direction of Computer Science Curriculum in K-12 Education* (Doctoral dissertation, The University of Western Ontario (Canada)).

Fluck, A., Webb, M., Cox, M., Angeli, C., Malyn-Smith, J., Voogt, J., & Zagami, J. (2016). Arguing for computer science in the school curriculum. *Journal of Educational Technology & Society*, *19*(3), 38-46.

Fosnot, C. T. (1996). *Constructivism: Theory, perspectives, and practice*. Teachers College Press.

Gadanidis, G., Brodie, I., Minniti, L., & Silver, B. (2017a). Computer coding in the K-8 mathematics curriculum? *What works: Research into practice*, 69, 1-4.

Gadanidis, G., Hughes, J. M., Namukasa, I., & Scucuglia, R. (2019). Computational modelling in elementary mathematics teacher education. In S. Llinares & O. Chapman (Eds.), *International Handbook of Mathematics Teacher Education: Volume 2* (pp. 197-222). Brill Sense.

Gadanidis, G., Hughes, J. M., Minniti, L., & White, B. J. (2017). Computational thinking, grade 1 students and the binomial theorem. *Digital Experiences in Mathematics Education, 3*(2), 77–96. https://doi.org/10.1007/s40751-016-0019-3

Gannon, S., & Buteau, C. (2018). Integration of Computational thinking in Canadian provinces. In *Online Proceedings of the Computational Thinking in Mathematics Education Symposium*.

Government of Northwest Territories. (2021). *Frequently asked questions: NWT partnering with British Columbia for JK-12 school curriculum*. https://www.ece.gov.nt.ca/sites/ece/files/resources/2021-11_-_faq_-_nwt_to_adopt_bcs_jk-12_curriculum_-_english_-_final.pdf

Government of Yukon. (2022). *Learn about the Yukon's school curriculum*. https://yukon.ca/en/school-curriculum

Gravel, B. E., & Wilkerson, M. H. (2017). Integrating computational artifacts into the multi-representational toolkit of physics education. In R. Duit, D. Treagust, & H. Fischer (Eds.), *Multiple Representations in Physics Education* (pp. 47–70). Springer. https://doi.org/10.1007/978-3-319-58914-5_3

Grover, S., & Pea, R. (2018). Computational thinking: A competency whose time has come. In S. Sentance, E. Barendsen, & C. Schulte (Eds.), *Computer science education: Perspectives on teaching and learning* (pp. 19–38). Bloomsbury Academic. https://doi.org/10.5040/9781350057142.ch-003

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

Harel, I. E., & Papert, S. E. (1991). *Constructionism*. Ablex Publishing.

Hennessey, E.J.V.,, Mueller, J., Beckett, D., & Fisher, P.A. (2017). Hiding in plain sight: Identifying computational thinking in the Ontario elementary school curriculum. *Journal of Curriculum and Teaching 6*(1), 79-96. https://doi.org/10.5430/jct.v6n1p79

Holbert, N., Berland, M., & Kafai, Y. B. (2020). Introduction: fifty years of constructionism. In N. Holbert, M. Berland, & Y.B. Kafai (Eds), *Designing constructionist futures: The art, theory, and practice of learning designs*, (pp. 1-16). MIT Press.

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

Information and Communications Council. (2017). *The next talent wave: Navigating the digital shift.* https://www.ictc-ctic.ca/wp-content/uploads/2017/04/ICTC_Outlook-2021.pdf

Kafai, Y. B. (2016). From computational thinking to computational participation in K-12 education. *Communications of the ACM, 59*(8), 26-27. https://doi.org/10.1145/2955114

Kafai, Y. B., & Proctor, C. (2022). A revaluation of computational thinking in K–12 education: Moving toward computational literacies. *Educational Researcher, 51*(2), 146-151. https://doi.org/10.3102/0013189X211057904

Khanlari, A. (2013). Effects of educational robots on learning STEM and on students' attitude toward STEM. In *2013 IEEE 5th Conference on Engineering Education* (pp. 62–66). https://doi.org/10.1109/ICEED.2013.6908304.

Lee, C., & Soep, E. (2023). *Code for What?: Computer Science for Storytelling and Social Justice*. MIT Press.

Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., ... & Werner, L. (2011). Computational thinking for youth in practice. *ACM Inroads*, *2*(1), 32–37. https://doi.org/10.1145/1929887.1929902

Merriam, S. B., & Tisdell, E. J. (2015). *Qualitative research: A guide to design and implementation*. John Wiley & Sons.

Milton, P. (2015). *Shifting Minds 3.0: Redefining the Learning Landscape in Canada*. C21 Canada.

New Brunswick Department of Education and Early Childhood Development. (2016). *Middle School Technology Education*. https://www2.gnb.ca/content/dam/gnb/Departments/ed/pdf/K12/curric/TechnologyVocational/Middle%20School%20Technology.pdf

Newfoundland and Labrador Department of Education. (2002). *Technology education: Communications technology module grade 7.* https://www.gov.nl.ca/education/files/k12_curriculum_guides_teched_gr7_g7_comm-module_june2002.pdf

Newfoundland and Labrador Department of Education. (2006). *Technology education: Control technology module 8.* https://www.gov.nl.ca/education/files/k12_curriculum_guides_teched_gr8ctrltech_g8control.pdf

Nova Scotia Department of Education and Early Childhood Development. (2016a). *Information and communication technology/Coding 4–6 integration*. https://www.ednet.ns.ca/files/curriculum/infotech_coding_4-6_streamlined.pdf

Nova Scotia Department of Education and Early Childhood Development. (2016b). *Nova Scotia's action plan for education: Annual report 2016.* https://www.ednet.ns.ca/docs/actionplan-annualreport-2016.pdf

Nunavut Department of Education. (2019). 2019 – 2020 Nunavut Approved Curriculum and Teaching Resources. https://gov.nu.ca/sites/default/files/2019-20_nunavut_approved_curriculum_and_teaching_resources.pdf

Ontario Ministry of Education. (1983). *Computer studies: Intermediate and Senior Division*.

Ontario Ministry of Education. (2008). *The Ontario curriculum grade 10 to 12: Computer studies*. http://www.edu.gov.on.ca/eng/curriculum/secondary/computer10to12_2008.pdf

Ontario Ministry of Education. (2020). *The Ontario curriculum grades 1–8: Mathematics*. https://www.dcp.edu.gov.on.ca/en/curriculum/elementary-mathematics/downloads

Ontario Ministry of Education. (2022). *The Ontario curriculum grades 1–8: Science and Technology*. https://www.dcp.edu.gov.on.ca/en/curriculum/science-technology/downloads

Papavlasopoulou, S., Giannakos, M. N., & Jaccheri, L. (2019). Exploring children's learning experience in constructionism-based coding activities through design-based research. *Computers in Human Behavior* (99), 415-427. https://doi.org/10.1016/j.chb.2019.01.008

Papert, S. (1993). *Mindstorms: Children, computers, and powerful ideas* (2nd ed.). Basic Books.

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

Popat, S., & Starkey, L. (2019). Learning to code or coding to learn? A systematic review. *Computers & Education*, *128*, 365-376. https://doi.org/10.1016/j.compedu.2018.10.005

Québec Ministère de l'Éducation. (2001). Québec education program: Preschool education, elementary education. http://www.education.gouv.qc.ca/fileadmin/site_web/documents/education/jeunes/pfeq/PFEQ_presentation-primaire_EN.pdf

Québec Ministère de l'Éducation. (2009). *Progression of learning: Science and technology*. http://www.education.gouv.qc.ca/fileadmin/site_web/documents/education/jeunes/pfeq/PDA_PFEQ_science-technologie-primaire_2009_EN.pdf

Resnick, M. (2018, September 16). Computational Fluency. *Medium*. https://mres.medium.com/computational-fluency-776143c8d725

Sengupta, P., Kinnebrew, J. S., Basu, S., Biswas, G., & Clark, D. (2013). Integrating computational thinking with K-12 science education using agent-based computation: A theoretical framework. *Education and Information Technologies, 18*(2), 351-380. https://doi.org/10.1007/s10639-012-9240-x

Smith, M. (2015, January 3). Computer Science For All. The White House: President Barack Obama. https://obamawhitehouse.archives.gov/blog/2016/01/30/computer-science-all

Sullivan, F. R., & Heffernan, J. (2016). Robotic Construction Kits as Computational Manipulatives for Learning in the STEM Disciplines. *Journal of Research on Technology in Education,* 48, 1–24. https://doi.org/10.1080/15391523.2016.1146563.

Tissenbaum, M., Sheldon, J., & Abelson, H. (2019). From computational thinking to computational action. *Communications of the ACM, 62*(3), 34-36. https://doi.org/10.1145/3265747

Tissenbaum, M., Weintrop, D., Holbert, N., & Clegg, T. (2021). The case for alternative endpoints in computing education. British Journal of Educational Technology, 52(3), 1164-1177.

Vogel, S., Santo, R., & Ching, D. (2017). Visions of computer science education: Unpacking arguments for and projected impacts of CS4All initiatives. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education* (pp. 609–614). ACM. https://doi.org/10.1145/3017680.3017755

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

Webb, M. E., Cox, M. J., Fluck, A., Angeli-Valanides, C., Malyn-Smith, J., & Voogt, J. (2015). Thematic working group 9: curriculum-advancing understanding of the roles of computer science/informatics in the curriculum. In *Summary Report: Technology Advance Quality Learning for All* (pp. 60–69). EDUSummit.

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., & Wilensky, U. (2016). Defining computational thinking for mathematics and science classrooms. Journal of Science Education and Technology, 25(1), 127–147. https://doi.org/10.1007/s10956-015-9581-5

Wilkerson, M. H., Shareff, R., Laina, V., & Gravel, B. (2018). Epistemic gameplay and discovery in computational model-based inquiry activities. *Instructional Science, 46*(1), 35-60. https://doi.org/10.1007/s11251-017-9430-4

Wilkerson, M. H. & Fenwick, M. (2017). The practice of using mathematics and computational thinking. In C. V. Schwarz, C. Passmore, & B. J. Reiser (Eds.), *Helping Students Make Sense of the World Using Next Generation Science and Engineering Practices*. National Science Teachers' Association Press.

Wilkerson-Jerde, M. H, Gravel, B. E., & Macrander, C. (2015). Exploring shifts in middle school learners' modeling activity while generating drawings, animations, and computational simulations of molecular diffusion. *Journal of Science and Educational Technology, 24* (2-3), 396-415. https://doi.org/10.1007/s10956-014-9497-5

Wing, J. (2006). Computational Thinking. *Communications of the ACM, 49*(3), 33-35. https://doi.org/10.1145/1118178.1118215

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

Zhang, L., Kirschner, P. A., Cobern, W. W., & Sweller, J. (2022). There is an evidence crisis in science educational policy. *Educational Psychology Review, 34*(2), 1157-1176. https://doi.org/10.1007/s10648-021-09646-1

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.