The Past, Present, and Future Direction of Computer Science Curriculum in K-12 Education

Steven Floyd, *The University of Western Ontario*

Supervisor: Dr. George Gadanidis, *The University of Western Ontario*

Abstract

This integrated article thesis provides an analysis of the past, present, and potential future state of Computer Science (CS) in K-12 education. Once implemented in optional courses at the secondary level, CS concepts and skills are now being integrated into other subject areas such as mathematics, science, and technology and other grades including K-8. This new state of K-12 CS education is explored through an analysis of 1) related theory reflected in the literature, 2) historical secondary school CS curriculum, 3) enrolment data and important issues related to equity, diversity, and inclusion, and 4) K-8 CS-related curriculum approaches currently being implemented in educational jurisdictions across Canada. The four articles in this dissertation employ a qualitative approach to research, drawing on a constructivist epistemology. Thematic Analysis is used to examine the goals and rationale of historical curriculum documents from Ontario and Document Analysis is used to compare various K-8 curriculum documents from across Canada. Together, the chapters included in this integrated article thesis provide a comprehensive analysis of K-12 CS education that supports educators, policy makers, and researchers in the field during a transformative time.

Keywords

coding, computer science, computational thinking, computational literacy, K-12, education, curriculum, technology

Summary for Lay Audience

This integrated article dissertation explores the field of Computer Science (CS) education in kindergarten to grade 12 (K-12), during a transformative time. In the past, CS concepts and skills were introduced to students in optional courses at the high school level, but now these concepts and skills are being introduced into other subject areas, such as mathematics, science, and technology and other grades, such as K-8. This thesis explores this change by analyzing theoretical perspectives from a variety of researchers as well as historical CS-related curricula. Enrolment data in CS courses is also explored, and the important themes of equity, diversity, and inclusivity in CS education are investigated, as well new CS-related curricula in the K-8 grades. The goal of this dissertation is to develop an understanding of the past, present, and future direction of CS in K-12 education.

Dedication

To Lisa, Cohen, Maxwell, Charles and Elizabeth.

Acknowledgments

A heartfelt thank you to my wife Lisa, you have endured my early morning and late-night exploratory speeches on all things coding, CS, and CT and you have shared with me quite a few of your own. Since we both started teaching CS courses back in 2003, you have been an inspiration and an honest and thought-provoking critical lens throughout. You encouraged me to share my experiences in CS education with others, outside of my classroom, and without your support this thesis would not have even begun. You also just happen to be a wonderful wife and a wonderful mother to Cohen, Maxwell, Charles, and Elizabeth.

To Cohen, Maxwell, and Charles: thank you for testing out so many programming activities, and for listening to so many conversations about curriculum. To Elizabeth, thank you in advance for hopefully doing the same.

To Dr. George Gadanidis, supervisor extraordinaire. Thank you for your expertise and guidance in personal and professional matters throughout the PhD process. I feel as though you provided just what was needed, at each step along this journey.

To Dr. Immaculate Namukasa, thank you for your continued support throughout the last few years, your feedback and perspectives have improved this work significantly. Also, thank you to Beckett Smith, a student whom I will never forget and one who continues to motivate me to share the power and wonder of technology with students.

To the educators at Mother Teresa Catholic Secondary School and at the LDCSB, those involved in developing and writing e-learning courses for TELO, and all those at TVO and at Ontario’s Ministry of Education: your insights and perspectives have helped shape my view of learning, research, and teaching and I suspect, have also helped shape this thesis.

My journey in computers and education began in the 1980s when my father, Derek, would bring home Pets, Commodores and programmable turtles. He introduced me to the idea that a computer can be a transformative force in learning, teaching, and thinking. My mother, Yvonne, was a kindergarten teacher and her approach to teaching in the younger grades embodied critical thinking, 21st century learning, and independence, long before these became popular topics. Thank you, I’m forever grateful to you both.

Table of Contents

List of Tables

List of Figures

List of Appendices

Preface

This thesis has been developed as an integrated article dissertation that includes an introduction, four main articles, and an integrated chapter connecting the main ideas and findings. These sections are briefly described below.

Chapter 1 is an introductory chapter that presents the context for the study, including a description of the problem being investigated, the purpose of the study, the research questions, related terms and definitions, and background on the researcher.

Chapter 2 analyzes existing literature from the field and provides a summary of the theoretical perspectives related to coding and computational thinking in K-12 education. Parts of this chapter have been published in the Proceedings of International Conference on Computational Thinking Education 2020. Appendix A includes a letter from the publisher, granting permission to include published parts in this dissertation.

Chapter 3 employs Thematic Analysis to investigate historical computer science curriculum implemented as an optional, isolated subject in secondary education. Preliminary work from this chapter was presented at the Fields Institute for Research in Mathematical Sciences and the Special Interest Group on Computer Science Education 2018 Conference. It was also published, in part, in The Math Knowledge Network Quarterly. Much of this work has also been accepted as a single-paper presentation at the Canadian Society for the Study of Education 2022 conference.

Chapter 4 investigates enrolment data related to CS courses implemented as optional credits in secondary schools in Ontario and explores important issues related to equity, diversity, and inclusivity in CS K-12 education. Parts of this chapter were published as a book chapter in the Handbook of Research on Equity in Computer Science in P-16 Education. Appendix C includes a letter from the publisher, granting permission to include the chapter in this dissertation. Preliminary work from this chapter was presented at the 2019 ACM Conference on International Computing Education Research in Toronto, Ontario.

Chapter 5 uses document analysis to provide a comparative analysis of existing coding curriculum in K-8 curricula from jurisdictions across Canada. This work has also been accepted as a single-paper presentation at the Canadian Society for the Study of Education 2022 conference.

Chapter 6 integrates findings from the four preceding chapters and demonstrates how intersecting themes from each chapter provide a picture of the current state of K-12 CS education and considers its past, present and potential future direction.

Together, these articles provide an analysis of the CS landscape in K-12 education at a transformational time. I am the sole author of all chapters, conference presentations, and articles that have been included.

Chapter 1

1 Introduction

This chapter introduces the problem, purpose of study and research questions that frame this integrated article dissertation. Relevant terms and definitions are also identified and described for the reader, as well as the background and positionality of the researcher.

1.1 Problem Description

The impetus for research focusing on Computer Science (CS) curriculum is the increased popularity of initiatives that attempt to broaden participation in CS education across the K-12 grades. This trend is exemplified by the programs, established by governments in the United States and Canada, to provide all students with an opportunity to learn to program a computer. In the United States, 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, 2016). 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). These initiatives demonstrate a recognition, on behalf of governments, of the importance of broadening participation in CS education and often reflect an economic argument maintaining that the knowledge and skills related to CS will be critical in the workforce of the future (Passey, 2017). Historical and contemporary theoretical perspectives from the field; however, present several alternative motivations for the broadening of CS-related education to all. In the 1970s, Seymour Papert introduced the K-12 education field to the idea that a computer could fundamentally change education by serving as a "tool to think with" (Papert, 1993). Decades later, in 2006, Jeanette Wing popularized a different approach, that argued that all students should program a computer in order to think like a computer scientist, through the development of Computational Thinking (Wing, 2006). Since that time, a number of researchers have provided additional detail and direction for Wing's (2006) Computational Thinking, while others have proposed other approaches that include Computational Action (Tissenbaum et al, 2019), Computational Fluency (Resnick, 2018), Computational Literacy (diSessa, 2018), and Computational Participation (Kafai, 2016).

Currently, this variety of perspectives is not well understood and even the general idea that all students should learn to program a computer is contentious (Webb et al, 2020). Considering the number of educational jurisdictions beginning to integrate CS-related concepts and skills in other subjects and grades, it is important to analyze how this is being done, what approaches and directions are being represented in new curriculum, and how this new curriculum might impact the more traditional implementation of CS education. In addition, as these concepts and skills are expanded to all learners, it is important to develop an understanding of potential equity, diversity, and inclusivity issues apparent in the traditional delivery model of CS education. This can help determine what can be done to alleviate these concerns or ensure that they are not reproduced as implementation models change.

1.2 Purpose of the Study

The purpose of this study is to develop an understanding of the current, evolving state of K-12 CS education by providing: 1) an analysis of literature that reflects theoretical perspectives in the field of CS K-12 education, 2) an understanding of the historically optional nature of CS education in terms of its placement in curriculum, goals and specific components, 3) an analysis of enrollment patterns and related issues including those concerning equity, diversity, and inclusivity, and 4) a comparative analysis of current approaches to CS curriculum in the K-8 grades that is based on the research literature.

1.3 Research Questions

This integrated article dissertation will provide an analysis of K-12 CS education through the lenses of theoretical perspectives, enrolment, and curriculum. To do so, the following question will be answered:

1. What is the current, and potentially future, direction of CS in K-12 education?

This will be answered by focusing on the following sub-questions:

a. What are the theoretical approaches presented in the literature that relate to the integration of CS concepts and skills in the K-12 grades?

b. What do curriculum documents reveal about the nature of historical CS K-12 education in terms of goals, rationale, and implementation models?

c. What do enrolment patterns reveal about the nature of historical CS K-12 education in terms of equity, diversity, and inclusivity?

d. What are the CS-related concepts and skills currently found in Canadian, K-8 provincial curricula and how do these reflect theoretical perspectives and historical CS K-12 education goals and rationale?

1.4 Terms and Definitions

1.4.1 Curriculum

A key focus for this study is the analysis and discussion surrounding curriculum. The term curriculum can have several different meanings, depending on the context or even the jurisdiction in which it is used. For some, curriculum can mean the activities and lesson plans developed for a class, while for others curriculum might mean the learning expectations and standards that students must meet. Curriculum may also be classified in terms of being either formal, that which is public and officially recognized, and actual, that which is carried out in the classrooms (Portelli, 1993). The learned curriculum is quite simply what students actually learn (Moercke & Eika, 2002) while critical curriculum theorists often refer to the hidden curriculum, first identified by Philip Jackson (1968), as one that is implicit and that rewards certain values, dispositions, and social and behavioural expectations. A categorization that is useful in this study is provided by Doyle (1992), who defines three levels of curriculum as either institutional, programmatic, or classroom. The institutional curriculum is broad, general, and abstract representing belief systems of an ideal educational experience while in contrast, the classroom curriculum involves the learning experiences that arise as teachers engage with students within the schools (Deng, 2010). Sometimes considered to be in the middle of these two extremes is the programmatic curriculum, which involves formal organizational structures such as school subjects and courses of study, and is enacted through policy documents, syllabi, and textbooks (Deng, 2010). The emphasis of this study is on the programmatic curriculum, with a specific focus on the policy documents developed by educational jurisdictions to communicate the expectations and outcomes related to CS concepts and skills.

1.4.2 Computer Science Related Terms

Like any field of study, the definitions of key terms within CS education are sometimes contentious. This section of the introductory chapter does not aim to resolve these disagreements, instead it is intended to explain the use of terms within this work.

Computer science, computer programming, coding, and computational thinking are related terms that appear in the literature and curricula and it is difficult to make clear and precise distinctions between them. For the sake of this dissertation, the broad term "computer science" will be used extensively in order to capture a number of these related concepts and skills. The term "coding" will be used in certain sections when referring to specific CS-related concepts and skills arising in K-8 curricula, as this appears to be a popular use of the term in policy documents (Government of Canada, 2019). The term Computational Thinking is explored in depth in Chapter 2. When Computational Thinking appears outside of Chapter 2, the definition in use is that of Aho's (2012) which states that "computational thinking is the thought processes involved in formulating problems so their solutions can be represented as computational steps and algorithms" (p. 832). This definition is recommended by Denning (2017) as it "captures the spirit of computational thinking expressed over 60 years of CS and 30 years of computational science. It also captures the spirit of computational thinking in other fields such as humanities, law, and medicine" (p. 35). Additionally, specific terms will be used when it is important to represent how an organization or educational jurisdiction refers to specific initiatives. As an example, in 2016, the Ontario Ministry of Education announced plans to support elementary teachers in integrating "coding and computational thinking" skills into their teaching (Ontario Ministry of Education, 2016). These terms were specifically used in the government's announcement, and this specificity is important to capture within the dissertation.

1.5 Organization of the Study

This dissertation is written as an integrated article thesis that comprises this introductory chapter, four main works (chapters 2, 3, 4 and 5) and a final integrated chapter that connects the main works. Together these six chapters provide context for the study and answer the research questions posed in the Research Questions section above.

1.6 Background and Positionality of the Researcher

Considering the qualitative nature of the work and the role of the researcher as a tool in the investigation, how the researcher situates themselves within the study can potentially impact data collection, data analysis and findings (Merriam & Tisdell, 2015). It is therefore important to clearly state the positionality of the researcher, as well as their experience in the chosen field of study.

I began taking university CS courses in September 1997, where I reached the conclusion that this subject could be taught in a creative and engaging way and was something that had the potential to appeal to all students. In 2003, I started my career as a CS teacher and 15 years later I was awarded the 2017 Computer Science Teachers Association Award for Teaching Excellence, presented by Infosys Foundation USA, the Association for Computing Machinery and the Computer Science Teachers Association. During my time as a secondary CS and Computer Engineering teacher, I led action research projects related to CS education with students and teachers in the elementary grades. I have worked as an independent consultant in the area of CS integration, a high school CS online course writer, a Bachelor of Education instructor in Computational Thinking in Mathematics and Science Education, and I am currently an Education Officer with Ontario's Ministry of Education, working in the area of STEM curriculum and policy development.

My interest in curriculum and policy began when completing a Masters in Educational Policy. While I am aware that a number of factors impact student learning, including specific resources, classroom activities, and teacher professional development, I believe that what is and is not included in mandatory curriculum has a dramatic impact on the learning of students, and can reveal important information related to the goals and perspectives of policy makers and governments. I recognize the importance and power of curriculum and educational policy and I appreciate the opportunity to have an impact in this area.

This qualitative research has been approached from a constructivist epistemology, which embodies underlining assumptions that include the following:

- we construct meanings for ourselves, as we interpret the world;
- we engage with and make sense of the world based on our historical and social perspectives;
- the generation of findings and meaning is social, arising from interactions with, or artifacts from, the human community (Crotty, 1998).

My teaching and classroom activities embody Papert's Constructionist learning theory, that shares Constructivism's idea of building our own knowledge structures, but also focuses attention on the importance of constructing a "public entity" (Harel & Papert, 1991). In my CS and computer engineering classes, these public entities often took the form of software solutions and physical computing artifacts that connected learning to a number of cross-curricular contexts. Examples include small software applications that analyzed sports data, the development of small computer games, robotics projects, and interactive, programmed art with LEDs and programmed musical tones. After years of working at the intersection of education and technology, I see the computer, the programming environments, and the physical computing devices in much the same way that Seymour Papert saw his childhood physical gears and his famous Logo Turtle, as "objects to think with" (Papert, 1993, p. 11).

Finally, and perhaps most importantly, I would like to acknowledge the themes of equity, diversity, and inclusivity that appear within this thesis, and to explain my current and continued allyship with related initiatives. The broadening of CS education to support underrepresented groups has consistently been a goal of my work, and I have attempted to ensure that my efforts are supported by research and best practices. The focus of this research is on curriculum and a reason for this is my view that large scale, educational policy has the power to positively influence equity, diversity, and inclusivity concerns in CS education and to make this important area of learning more accessible to all students. I am aware that I do not identify as a member of an underrepresented group in CS, and as a result, I have been hesitant to take up space in this area, when perhaps it is not my voice that needs to be heard. My wife Lisa does extensive work in the area of K-12 CS education and research and I am grateful to be able to discuss with her these concerns, and better understand my role as an ally. The recent birth of our daughter has also provided me with a new experience and perspective, and I hope that these can also help me better understand equity, diversity, and inclusivity issues and allyship. I have discussed my role in researching and presenting equity, diversity, and inclusivity issues in CS education with a number of professors at Western University, and was humbled to have received the Canadian Research Centre on Inclusive Education Research Award in 2019. I continue to consider how I can help contribute positively to equity, diversity, and inclusivity initiatives in CS education. I hope that this thesis can have an impact in supporting positive changes in K-12 CS education.

1.7 Chapter References

Aho, A. V. (2012). Computation and computational thinking. *Computer Journal, 55*(7), 832-835. https://doi.org/10.1093/comjnl/bxs074

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

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. (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

Deng Z (2010). Curriculum planning and systems change. In P. Peterson, E. Baker, & B. McGaw (Eds.), *International Encyclopedia of Education* (pp. 384-389). Elsevier.

Doyle, W. (1992). Curriculum and pedagogy. In P. W. Jackson (Ed.), *Handbook of research on curriculum* (pp. 486–516). Macmillan.

Government of Canada. (2019). *CanCode*. https://www.ic.gc.ca/eic/site/121.nsf/eng/home

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

Jackson, P. W. (1968). *Life in classrooms*. Holt, Reinhart & Winston.

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

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

Moercke, A. M., & Eika, B. (2002). What are the clinical skills levels of newly graduated physicians? Self-assessment study of an intended curriculum identified by a Delphi process. *Medical education*, *36*(5), 472-478. https://doi.org/10.1046/j.1365-2923.2002.01208.x

Ontario Ministry of Education. (2016, December 5). Ontario helping students learn to code: New supports for coding and computational skills in Ontario schools. *Ontario Newsroom*. https://news.ontario.ca/en/release/42956/ontario-helping-students-learn-to-code

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

Portelli, J. P. (1993). Exposing the hidden curriculum. *Journal of curriculum studies, 25*(4), 343-358. https://doi.org/10.1080/0022027930250404

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

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

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

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

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

Chapter 2

2 Theoretical Perspectives Related to Computer Science in K-12 Education

The integration of computer science (CS) related concepts and skills into areas outside of the traditional, CS high school courses is becoming an integral part of educational reforms in Ontario, across Canada, and internationally. In the spring of 2019, Canada's federal government announced an additional $60 million of funding for their CanCode coding initiative (Department of Finance Canada, 2019) while in spring 2020, Ontario released curriculum that included coding expectations in grades 1-8 Mathematics (Ontario Ministry of Education, 2020). Across Canada, Alberta, British Columbia, New Brunswick, and Nova Scotia have recently included CS concepts and skills in their current or draft K-8 curricula while beyond Canada the integration of CS into K-8 education has become an international phenomenon (Gadanidis et al., 2017). As the broadening of CS education continues, it is important to situate these initiatives within the research literature and theory related to CS education in the K-12 grades.

2.1 Introduction

It has been argued that the understanding of CS concepts contributes to the development of important technical skills that form the basis of a number of lucrative, high-status and flexible careers within the continually growing field of technology (Information and Communications Technology Council, 2017). In addition, CS concepts and the thought processes involved in Computational Thinking (CT) are discussed as being valuable for all students to learn (Wing, 2006) as they are applicable to a wide range of careers. Coding skills have also been recognized as a new type of literacy (diSessa, 2018), a form of personal expression (Brennan & Resnick, 2012) and a critical part of being an educated, 21st century citizen (Margolis et al., 2012). What follows is a literature review of these and other theoretical perspectives related to the broadening of CS education and the potential for integration of CS concepts and skills outside of the traditional, high school CS classroom. The analysis begins with an introduction to Wing's idea of CT and then presents perspectives from other researchers, including their concerns with Wing's approach. This chapter is meant to provide an understanding of foundational, theoretical approaches to situate the rest of this thesis and to provide context for contemporary approaches to the broadening of CS in K-12 education.

2.2 Wing's Idea of Computational Thinking

In March of 2006, the Communications of the ACM published an article by Wing entitled Computational Thinking. At the time of publishing, Wing was the head of the Computer Science Department at Carnegie Mellon University and was seeking to expand the scope of CS education beyond the post-secondary levels. In Computational Thinking, Wing articulated the characteristics and importance of a "universally applicable attitude and skill set" (p. 33) called Computational Thinking (CT), that goes beyond simply coding a computer and instead involves thinking like a computer scientist. She also encouraged the CS community to inspire the public's interest in the field of CS and expose all K-12 students to computational methods and models in an effort to make CT commonplace.

Wing initially defined CT as "solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science" (Wing, 2006, p. 33). Later, in 2011, she refined her definition to the "thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent" (p. 20).

While researchers and educators have discussed Wing's initial definition at length, they have also criticized her focus on problem-solving and thinking like a computer scientist. Lorena Barba (2016) explains that Wing's view fails to acknowledge CT as "a source of power to do something and figure things out, in a dance between the computer and our thoughts". Barba goes on to explain that viewing the computer as a formal tool to understand, and then apply to a problem later, takes away its power:

> The operational aspect of making problems computable is essential, but not aspirational. Most people don't want to be a computer scientist, but everyone can use computers as an extension of our minds, to experience the world and create things that matter to us. (para. 23)

Barba was attempting to move discussions away from a CS-centric CT and move towards an idea of computational learning that would allow students to use computing as a means to create new knowledge in a broad number of domains. diSessa (2018) shares similar views, as he notes that Wing's position appears firmly entrenched in the discipline of CS, but for something to have as broad of aspirations as CT, it cannot belong to one discipline. He also notes that Wing's CT view fails to recognize foundational literature in the field, including work from Papert, who like himself, aimed to bring computational ideas to the wider population "for general intellectual purposes" (diSessa, 2018, p. 27). These computational ideas and the general theory for the broadening of CS will be further explored, beginning with the historical and foundational work from Papert.

2.3 Papert and Constructionism

Described as the father of educational computing (Stager, 2016), Papert and his ideas were foundational in terms of considering the learning and teaching that takes place with computers (Kafai & Burke, 2014). With a bachelor's degree in philosophy and a PhD in mathematics, Papert became a research associate at MIT in 1964 and a professor in 1969. Before arriving at MIT, Papert worked closely with Piaget, whose theory of cognitive development heavily influenced Papert's work: "I take from Jean Piaget a model of children as builders of their own intellectual structures" (Papert, 1993, p. 7). Papert built on Piaget's theory of constructivism by developing his own theory of learning that he called constructionism (Stager, 2016). Both theories focus on learning being an active process of constructing knowledge and both include the idea that children learn new concepts by relating them to things that they already know (Ames, 2018). Where they differ; however, is how Papert acknowledges the importance of culture as the source of the materials that students will use to build their knowledge (Papert, 1993). Papert believed that in some cases the culture provides the learning materials in abundance, which facilitates Piagetian learning. In other cases, however, where there is a slower development of a particular concept, Piaget attributed this to greater complexity or formality, whereas Papert saw the critical factor as "the relative poverty of the culture in those materials that would make the concepts simple and concrete" (Papert, 1993, p. 7).

It was for this reason that Papert was so enamored with the computer as a learning tool. He felt that the relative poverty of a culture, school or classroom could be cured by a computer, which he called the Proteus of machines, that can “take on a thousand forms and can serve a thousand functions” (Papert, 1993, p. xxi). Papert’s research agenda at MIT was shaped by two major themes surrounding the computer and education: 1) children are capable of learning to use computers in masterful ways, and 2) learning to use computers can change the way that children learn other things (Papert, 1993). While the computer played a central role in his work with children, 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).

At MIT, Papert developed the Logo computer programming language, which he felt could alter the relationship that students had with computers. Rather than having students be programmed by a computer (through computer-based exercises, computer-based feedback or by having the computer dispense information), the Logo programming environment reversed this relationship by having the student program the computer itself, which essentially meant teaching the computer how to think. Papert felt that in teaching the computer how to think, the student would begin to consider how they themselves think: “The experience can be heady: Thinking about thinking turns the child into an epistemologist, an experience not even shared by most adults” (Papert, 1993, p. 19). When further describing the epistemological nature of children’s work with Logo, Papert comes very close to describing a modern form of CT:

> I have invented new ways to take educational advantage of the opportunities to master the art of deliberately thinking like a computer, according, for example, to the stereotype of a computer program that proceeds in a step-by-step, literal, mechanical fashion. There are situations where this style of thinking is appropriate and useful. Some children’s difficulties in learning formal subjects such as grammar or mathematics derive from their inability to see the point of such a style. (Papert, 1993, p. 27)

Papert uses the term "mechanical thinking" (Papert, 1993, p. 27) to describe the type of thinking that students are introduced to when programming in Logo. He emphasizes that by introducing students to mechanical thinking, they suddenly become aware that there is such a thing as a thinking style, and they begin to consider other thinking styles that might exist, as well as how and why they might choose between styles. Papert first uses the term "computational thinking" (p. 182) when he discusses how the visions of early experiments were insufficiently developed, in terms of how to integrate this type of thinking into everyday life:

> In most cases, although the experiments have been interesting and exciting, they have failed to make it because they were too primitive. Their computers simply did not have the power needed for the most engaging and shareable kinds of activities. Their visions of how to integrate *computational thinking* [emphasis added] into everyday life was insufficiently developed. But there will be more tries, and more and more. And eventually, somewhere, all the pieces will come together and it will catch. One can be confident of this because such attempts will not be isolated experiments operated by researchers who may run out of funds or simply become disillusioned and quit. They will be manifestations of a social movement of people interested in personal computation, interested in their own children, and interested in education. (p. 182)

Papert's work surrounding computers and education, and his development of the Logo programming language, sowed the seeds of this educational movement. Resnick, a former student of Papert's, exclaimed that he would be happy to spend the rest of his life nurturing these seeds (Resnick, 2020).

2.4 Resnick and Computational Fluency

Resnick is currently the LEGO Papert Professor of Learning Research at MIT Media Lab and the director of the MIT Lifelong Kindergarten research group that developed Scratch, the world's leading coding platform for kids. Resnick explains that Scratch was deeply inspired by Papert's Logo programming language but "goes beyond Logo by making programming more tinkerable, more meaningful, and more social" (Resnick, 2014, p.

14). While Resnick makes these claims, it is important to note specific distinctions between the Logo and Scratch programming languages. Papert’s Logo remained simple and focused specifically on mathematics, whereas Scratch has broader goals and many additional features, including a wide variety of programmable characters (sprites) and backgrounds and the ability to share projects online. While this has the potential to keep students engaged, there is also the possibility that additional features can distract students and possibly teachers from the learning of mathematics. Benton et al. (2016) explain that with carefully designed activities and pedagogy, such as their ScratchMath program, Scratch can be used effectively to support mathematics instruction.

In terms of Resnick’s approach to CT and its role in children’s education, he acknowledges, with co-author Brennan, that there is little agreement about what CT encompasses, and even less agreement about strategies for assessing CT (Brennan & Resnick, 2012). In order to provide further depth and clarity, they propose a CT framework that includes three key dimensions: computational thinking concepts, computational thinking practices and computational thinking perspectives.

Resnick and Brennan’s CT framework includes the concepts that designers engage in as they program. These include sequences, loops, parallelism, events, conditionals, operators, and data. CT practices differ to CT concepts in that the practices describe the processes of construction that student engage in while creating Scratch projects. The practices include being incremental and iterative, testing and debugging, reusing and remixing, and abstracting and modularizing. CT perspectives, which describe the evolving understanding that students using Scratch exhibit about themselves, their relationship to others, and the technological world include expressing, connecting, and questioning. Together, the concepts, practices and perspectives provide a broader understanding of CT. Resnick later articulated this broader understanding using his term Computational Fluency (Resnick, 2017).

The impetus for Resnick’s Computational Fluency was an attempt to focus on children developing as computational creators as well as computational thinkers (Resnick, 2017). Computational Fluency goes beyond computational concepts and problem-solving strategies of CT by including student's creativity and expression with digital tools (Resnick, 2017). When describing Computational Fluency, Resnick is quick to point out his emphasis on projects rather than puzzles:

> Most introductions to coding are based on puzzles. Kids are asked to create a program to move a virtual character past some obstacles to reach a goal. With Scratch, we focus on projects instead of puzzles. When we introduce kids to Scratch, we encourage them to create their own interactive stories, games and animations. (Resnick, 2017, p. 48)

Resnick (2017) acknowledges Wing's view of CT and its impact on the development of thinking skills, but claims that becoming fluent, either in traditional writing or in code, helps a student to move beyond CT thinking skills by also developing a voice and an identity. While carefully constructed puzzles may help with fostering CT skills, Resnick (2017) believes that the broadening of CS education should allow students to develop their voice by learning to express themselves in new ways and by incorporating coding into everyday life. In terms of programming projects and their role in developing an identity, Resnick (2017) shares the following:

> In today's society, digital technologies are a symbol of possibility and progress. When children learn to use digital technologies to express themselves and share their ideas through coding, they begin to see themselves in new ways. They begin to see the possibility for contributing actively to society. They begin to see themselves as part of the future. (p. 50)

Resnick's emphasis on having students design digital artifacts is well grounded in Papert's constructionist approach to learning. He acknowledges the surge of interest in coding "provides an opportunity for reinvigorating and revalidating the Constructionist tradition in education" (Resnick, 2014, p. 7). Kafai, another one of Papert's students, acknowledges the importance of sharing and collaboration in the broadening of CS, and these components are embodied in her extension of CT that she calls Computational Participation (Kafai, 2016).

2.5 Kafai and Computational Participation

Kafai is currently the Lori and Michael Milken President's Distinguished Professor in the Graduate School of Education at the University of Pennsylvania. Kafai attended graduate school at Harvard University and was part of the team that, along with Resnick, helped developed the Scratch programming language. Kafai’s work, while acknowledging the technical and tool-oriented approaches to coding, focusses much more on the social and participatory dimensions (Kafai et al., 2011; Kafai & Burke, 2013; Kafai et al., 2014). Kafai, and co-author Burke, discuss coding in terms of four dimensions characteristic of Papert’s Constructionist thought (social, personal, cultural, and tangible) and explain how these dimensions have evolved resulting in a new form of programming whereby students can create applications as part of a larger community (Kafai & Burke, 2014). These shared applications are the “public entity” that Papert and Harel (2002) describe as the important addition that Constructionism provides, as the building of knowledge structures “happens especially felicitously in a context where the learner is consciously engaged in constructing a public entity, whether it's a sand castle on the beach or a theory of the universe” (p. 2). This programming, as a participatory process, differs from Wing’s CT approach, in recognizing that “when code is created, it has both personal value and value for sharing with others” (Kafai & Burke, 2014, p. 17). Kafai (2016) argues that CT needs to be reframed as Computational Participation moving us “beyond tools and code to community and context” (p. 27).

Computational Participation acknowledges that CT is a social practice with a broad reach. Rather than an abstract discipline, programming is now a way to “make and be” in the digital world (Kafai, 2016, p. 27). Digital technologies are used for functional, political, and personal reasons and therefore all students should develop an understanding of interfaces, technologies, and systems that they encounter on a daily basis. By developing an understanding of these systems, students can fully participate in digital activities and social practices.

Computational Participation takes a broad view of computing and acknowledges its potential impact across a wide range of fields. This broad view shares some characteristics with Computational Literacy (CL), an idea that was developed by diSessa (2000) before Wing's ideas about CT became popular.

2.6 diSessa and Computational Literacy

diSessa is the Corey Professor of Education at Berkeley's Graduate School of Education where he researches forms of knowledge in physics, as well as the use of computer systems in teaching and learning. He started his work in computing education as a member of Papert's Logo group at the MIT Artificial Intelligence Laboratory and now focusses on the idea that computers can be the basis for a new form of literacy that is applicable to a wide variety of subjects, contexts, and domains (Weintrop et al., 2016).

diSessa (2018) imagines a world "in which computational knowledge – the prime example is programming – is as widely practiced as reading newspapers and novels is today" (Papert, 2006, p. 240). In presenting computing as a new form of literacy, diSessa advocated for the broad use of computers in schools, and for educators to see computing as means of transforming the teaching and learning of things that are hard for students to learn (Papert, 2006). diSessa uses algebra as an example of an epistemological entity that transformed complex and difficult ideas into a form "that is within the intellectual grasp of every competent high school student" (Papert, 2006, p. 241). He suggests that CL involves computing and computer programming concepts being integrated into school subjects in much the same way that algebra has become a tool in science, mathematics, and other subjects.

diSessa (2018) explains that his use of the term literacy goes beyond the idea of simply having a casual acquaintance with something. Instead, literacy means the adoption, by a broad group, or even a civilization, of a "particular infrastructural representational form for supporting intellectual activities" (diSessa, 2018, p. 4). diSessa criticizes the "computer science-centric" view in Wing's CT by acknowledging that because literacy is such a massive social and intellectual accomplishment, it can not belong to a single professional discipline. diSessa adds to this by providing practical advice:

> There is no single recipe for how computation changes a field or subfield. If your pursuits take you in different directions, then I suggest here, that will enrich the horizon for all of us. If they parallel or extend what I and others who are focused on the big picture have already done, perhaps we can converge sooner than might be expected. (diSessa, 2018, p. 28)

2.7 Denning, Aho, Wilkerson, Gadanidis and Modelling in Other Subject Areas

Denning is a Distinguished Professor of Computer Science at the Naval Postgraduate School in Monterey, California. He has worked extensively within the field of CS and CT and has published numerous works on computers and computing education. Denning (2017) explains that CT has been major component of CS since the 1950s and so has the idea that CT can benefit people in a variety of fields. Denning claims that recent attempts to make CT appealing to fields other than CS have led to “vague and confusing definitions of CT” (p. 33). Denning’s two main criticisms of Wing’s definition of CT include the absence of any mention of computational models as well as the suggestion that any sequence of steps constitutes an algorithm. He prefers, instead, to accept a definition of CT proposed by Alfred Aho (2012), which he claims better embodies the notion of CT from CS, computational science, as well as other fields such as the humanities, law, and medicine.

Aho is the Lawrence Gussman Professor Emeritus of Computer Science at Columbia University. Aho (2012) defined CT quite succinctly as “the thought processes involved in formulating problems so their solutions can be represented as computational steps and algorithms” (p. 832). Aho explained that an important part of the CT thought processes involves finding the appropriate models of computation, and if there are none, then developing new ones. This view is exemplified in some of the mathematical modelling work by Wilkerson (Wilkerson & Fenwick, 2017).

Wilkerson is an Assistant Professor in the Graduate School of Education at the University of California, Berkeley and with co-author Fenwick, suggests that CS shares language with mathematics that can be used to represent models resulting in a description of patterns and processes that can make up scientific and engineered systems (Wilkerson & Fenwick, 2017). Wilkerson and Fenwick (2017) note:

> While mathematics focuses on quantities, computational thinking focuses on processes. Students engaged in the practice of computational thinking break a complex problem or process up into smaller steps in order to better understand, describe, or explain it. It involves thinking about how computer tools and algorithms – specific instructions for how something should be done – can be used to make jobs like data collection and analysis or theory testing easier, more manageable, or more powerful. (Wilkerson & Fenwick, 2017, p. 189)

Wilkerson provides opportunities for students to use or build computational models and simulations in order to better understand scientific and engineered systems. An example of this work includes an investigation into groups of sixth-grade girls generating models of smell diffusion concepts using drawing, stop-motion animation, and computational simulations (Wilkerson-Jerde et al., 2015). The authors observed two modelling cycles that students engaged in, including a "messing about" modelling cycle, where ideas related to the spread of smell were described and represented together, and a "digging in" cycle where the computational simulation allowed the group to focus on testing and revising specific mechanisms that underlie smell. The authors concluded that this "digging in" cycle involved a more mechanistic focus that was facilitated by creating a computational object that encapsulated ideas from the "digging in" cycle. An additional example of this computational modelling and simulation work includes Wilkerson and co-authors acknowledging that the building of computational models supports structuring knowledge (e.g., mechanistic reasoning) and fostering reflection and refinement (e.g., modeling practices). An important caveat; however, was that that the modeling strategies that students engaged in was important, as the modelling strategies must be aligned with the modelling type that students are employing (Wilkerson et al., 2018).

Adding to the literature connecting CT, coding and mathematical modelling, is the work done by Gadanidis, a Professor in the Faculty of Education at Western University, in London, Canada. Gadanidis et al.'s (2019) research surrounding computational modelling, mathematics education and elementary teacher education reframes CT with a focus on what it can do (CT's affordances), rather than what it is (CT's definition). The authors identify ten affordances that come into play when modelling mathematical concepts and relationships with computational tools (Table 1) and explain that these affordances and the use of coding tools can create scenarios in which students and teachers can connect a variety of mathematical concepts together, in this case from different strands of the curriculum.

Table 1. Ten affordances of computational modelling (Gadanidis et al., 2019)

1. *Access:* computational modelling tools for young students have a low floor & a high ceiling, allowing use with minimum prerequisite knowledge and offering opportunities to investigate more complex relationships and concepts

2. *Agency:* a low floor, high ceiling access allows students conceptual freedom to investigate ideas and concepts of interest

3. *Abstraction:* the code used to develop computational models captures/abstracts essential characteristics and processes of concepts and relationships

4. *Tangible feel:* abstractions in computational models have a tangible feel as they can become objects of other code

5. *Automation:* computational models automate processes

6. *Dynamic modelling: a*utomation allows for dynamic modelling, where concepts and relationships can be modelled at the click of a button

7. *Surprise & insight:* parameters and other aspects of the code can be edited and modified, to explore other cases, and to offer opportunities for conceptual surprise and insight

8. *Audience:* computational models can easily be shared with others

9. *Re-use/Re-mix:* others can re-use shared computational models or re-mix them to create variations

10. *Performance:* digital media, inclusive of some coding environments are performative in their nature and allow users to not only write code, but to also insert multimodal text and tell stories through animation

These affordances and the use of coding tools can also promote the exploration of multiple mathematical processes such as problem solving, reasoning and proving, reflecting, computational strategies, representation and communication. In addition, mathematics concepts that may appear outside of the curriculum expectations for certain levels, can be explored as a result of the low floor, high ceiling nature of modelling with code (Gadanidis et al., 2019). The authors provide an example of trigonometry, previously introduced by Gadanidis (2012), where bar graphs reflecting the heights of hours on a clock form trigonometric graphs. An additional example of students exploring mathematics concepts that may appear outside of the curriculum expectations for certain levels includes grade 1 students being able to explore the rudiments of the Binomial Theorem through dynamic, computational modelling (Gadanidis et al., 2017).

The areas of investigation presented by Wilkerson and Gadanidis, related to computational simulations and models in science and mathematics, represent Grover's (2018) classification of integrating CT in an effort to enable or enrich learning in other disciplines.

2.8 Grover, a Tale of Two CTs and Consolidating Theory

Grover (2018), a computer scientist and learning sciences researcher based in Palo Alto, California, argues that in order to make sense of CT in K-12 education we need to distinguish between two main views. Grover's first view of CT is that of CS thinking in CS classrooms while her second is that of CT in other disciplines. She explains that ideally, students will get a chance to experience CT in both settings during their K-12 schooling. Grover also presents a brief timeline of CT starting with the problem-solving practices discussed by Forsythe (1967) and the elements of CS thinking discussed by Knuth (1980). In regard to Wing, Grover (2018) credits her definition of CT for igniting K-12 CS education and for calling attention to its role in other disciplines, but also acknowledges that there should not be a focus on CT changing everyday behaviors. Instead, CT should be viewed as playing a significant role in CS education and playing a role in helping students understand concepts within a variety of fields and disciplines.

This idea of understanding concepts in a variety of fields and disciplines is extended further in Table 2, as a means of organizing the theories within this chapter. Wing's CS-centric approach frames CS as a topic of study in and of itself, as students can think like a computer scientist, and draw on the fundamental CS concepts to solve problems, design systems, and understand human behaviour. Wing's CS focused approach has value for broadening the scope of the traditional CS classroom and encouraging participation in these courses. While Wing's initial work lacked description and depth, other researchers such as Brennan and Resnick (2012) and Grover and Pea (2018) have provided a more detailed description of CT components, which they call CT concepts, practices and perspectives.

Table 2. Theoretical approaches to the broadening of CS K-12 education

Theoretical Approach	Researcher	Details
CS as a topic of study in and of itself	Wing (2006, 2011)	*Computational Thinking* is a universally applicable skill set. Important for students to learn *to think like a computer scientist*. Solving problems, designing systems, and understanding human behaviour, by drawing on the *concepts fundamental to CS.* Studying CT is good for all students and allows them *to be better thinker*s.
CS-related concepts and skills as a tool in mathematics and science	Papert (1993)	Acknowledges students as builders of their own intellectual structures. The computer is a “tool to think with”. Coding can change the way students learn about other things. The computer serves as the Proteus of machines, providing the culture and materials that made the previous learning of concepts difficult.
	diSessa (2000, 2018)	*Computational Literacy* can transform the teaching and learning of things that are hard for students to learn. Like algebra, coding can be an epistemological entity that can transform complex and difficult ideas into a form that is within our intellectual grasp. Coding has the potential to be adopted as an infrastructural representational form for supporting intellectual activities.
	Barba (2016)	Avoid viewing the computer as a formal tool to understand, then apply to a problem later, as this takes away its power. Not everyone wants to be a computer scientist, but everyone can use computers as an extension of our minds. Potential for creating knowledge in a broad number of domains.
	Wilkerson et al. (2018) and Gadanidis et al. (2019)	Focus on *computational modelling of mathematics and scientific concepts*. Concepts are better understood through developing dynamic, computational representations.
CS concepts and skills for the social, personal, and cultural	Kafai (2016)	*Computational Participation* moves beyond tools and code, to community and context. Highlights the four dimensions of social, personal, cultural, and tangible. Coding as part of a participatory process and social practice, with a broad reach to a larger community. Code has personal value, and a value for sharing with others.
	Resnick (2017)	*Computational Fluency* highlights importance of children developing as creators, not just thinkers. Move away from puzzles and beyond concepts and problem solving, to creativity and expression. Focus on students creating a voice and identity. Recognizes the potential to reinvigorate and revalidate the constructionist ideals in schools.

The views of Papert (1993), Barba (2016), diSessa (2018), Wilkerson (Wilkerson et al., 2018), Gadanidis (Gadanidis et al., 2019), Kafai (2016) and Resnick (2017), while still appropriate in the CS classroom, are better suited than Wing's when one considers the disciplines outside of CS. These views embody a perspective of the computer as a tool rather than as an object of study in and of itself. Papert (1993), diSessa (2018), Barba (2016), Wilkerson (Wilkerson et al., 2018), and Gadanidis (Gadanidis et al., 2019) highlight the computer as a tool within mathematics and science, while Kafai (2016) and Resnick (2017) focus more on the social, personal and cultural affordances of programing the computer.

Wilkerson and Gadanidis's approach to having students use CS concepts and skills to build computational models and simulations in order to better understand mathematical, scientific and engineered systems is a powerful one for the mathematics and science classrooms. It means re-envisioning data collection, analysis, and theory testing, making it more manageable and providing younger students with the tools that experienced scientists and mathematicians use on a daily basis. Expanding this idea and aiming for an even greater impact, diSessa asks us to think big and orient ourselves to the best that can be imagined by presenting a model of how coding can potentially become a literacy. His idea of CL means a transformation in the way students learn mathematics, and he predicts that CL can dramatically overshadow the type of algebra and calculus literacy that students currently develop.

Kafai's Computational Participation and Resnick's Computational Fluency emphasize the idea that programming a computer is a social practice with functional, political, and personal value. This can provide meaningful context for programming projects, allowing them to be more closely connected to student's lives and communities. Most importantly, these components could also be beneficial in a broadening of CS to other subject areas and grades, as they include an emphasis on creativity and expression, which may invite coding activities within the Arts, Languages, or other contexts. If mathematics and science areas are the focus, however; it seems that Gadanidis, Wilkerson and diSessa's work would be most helpful as a starting point.

2.9 Conclusion

This analysis of the approaches to CS-related concepts and skills in K-12 education provides a theoretical context for further exploration of themes within this thesis. In Chapter 3, historical curriculum in Ontario is examined, and these theoretical frameworks can help identify whether or not a change of focus has occurred in terms of the concepts and skills included within secondary CS curricula. These frameworks will also prove valuable as new CS-related curriculum within the K-8 grades is explored in Chapter 5, as they can provide insight into the direction that various jurisdictions in Canada may have pursued to broaden CS education. In terms of the issues addressed in Chapter 4, including those of enrolment, equity, diversity, and inclusivity, the work within this chapter will help shed light on why students may or may not be attracted to secondary courses in CS, and may provide insight into how the broadening of CS could be made more or less effective, with the adoption or emphasis of a particular direction or approach. This foundational and theoretical chapter, in combination with the analysis of curriculum and enrolment in further chapters, will help present an understanding of the current field of CS and K-12 education, a field that was invigorated by Wing’s (2006) work, but that includes many more substantial and comprehensive theories upon which to situate itself in the coming years.

2.10 Chapter References

Aho, A. V. (2012). Computation and computational thinking. *Computer Journal, 55*(7), 832-835. https://doi.org/10.1093/comjnl/bxs074

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

Barba, L. (2016, March 15). Computational thinking: I do not think it means what you think it means. *Lorena A. Barba Group*. http://lorenabarba.com/blog/computational-thinking-i-do-not-think-it-means-what-you-think-it-means/.

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

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.

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

Forsythe, G. E. (1967). What to do till the computer scientist comes. *The American Mathematical Monthly, 75*(5), 454-462.

Gadanidis, G. (2012). Trigonometry in grade 3? *What Works? Research into Practice*, 42, 1-4.

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

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

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

Grover, S. (2018, November 5). A tale of two CTs (and a revised timeline for computational thinking) [Blog Post]. *Communications of the ACM*. https://cacm.acm.org/blogs/blog-cacm/232488-a-tale-of-two-cts-and-a-revised-timeline-for-computational-thinking/fulltext.

Grover, S., & Pea, R. (2017). 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://10.5040/9781350057142.ch-003

Government of Canada. (2019). *CanCode*. https://www.ic.gc.ca/eic/site/121.nsf/eng/home

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

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., & Burke, Q. (2013). The social turn in K-12 programming: Moving from computational thinking to computational participation. *ACM*, 603-608. https://doi.org/10.1145/2445196.2445373

Kafai, Y.B., & Burke, Q. (2014). *Connected code: Why children need to learn programming*. MIT Press.

Kafai, Y. B., Fields, D. A., & Searle, K. A. (2014). Electronic textiles as disruptive designs: Supporting and challenging maker activities in schools. *Harvard Educational Review, 84*(4), 532-563. https://doi.org/10.17763/haer.84.4.46m7372370214783

Kafai, Y. B., Peppler, K. A., Lemke, J., & Warschauer, M. (2011). Youth, technology, and DIY: Developing participatory competencies in creative media production. *Review of Research in Education, 35*(1), 89-119. https://doi.org/10.3102/0091732X10383211

Knuth, D.E. (1980). Algorithms in Modern Mathematics and Computer Science. *Stanford Department of Computer Science Report No. STAN-CS-80-786.*

Margolis, J., Ryoo, J. J., Sandoval, C. D., Lee, C., Goode, J., & Chapman, G. (2012). Beyond access: Broadening participation in high school computer science. *ACM Inroads, 3*(4), 72-78. https://doi.org/10.1145/2381083.2381102

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

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

Papert, S. (2006). Minding change. *Human Development, 49*(4), 239-247. https://doi.org/10.1159/000094373

Resnick, M. (2014). Give P's a chance: Projects, peers, passion, play. In G. Futschek, & C Kynigos (Eds), *Proceedings of the 3rd international constructionism conference 2014* (pp. 13-20). Austrian Computer Society.

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

Resnick, M. (2020, October 16). The seeds the Seymour Sowed. *Medium*. https://mres.medium.com/the-seeds-that-seymour-sowed-c2860379617b

Stager, G. S. (2016). Seymour Papert (1928-2016). *Nature, 537*(7620), 308-308. https://doi.org/10.1038/537308a

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. & 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

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

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

Wing, J. (2011). Research notebook: Computational thinking—What and why. *The link magazine*, *6*, 20-23.

Chapter 3

3 Historical Computer Science Curriculum: From 1966 to today

This chapter provides insight into the evolution and current state of secondary Computer Science (CS) education through an analysis of eight Ontario Ministry of Education documents that were released from 1966 to 2008. First, Thematic Analysis is used to compare the preambles of each document, providing insight into the intended goals and rationale for each curricula and how these have evolved over the years. After this analysis, a second section investigates the specific concepts and skills included in introductory, CS-related courses that have been implemented in Ontario. The results indicate that CS courses appeared first in Ontario's secondary education system within the Business and Commerce curriculum, with close connections to the data processing context. In addition, documents from as far back as 1966 and 1970 were clearly acknowledging many of the themes evident in today's discourse on the broadening of CS education. These include the economic argument for increasing CS participation, the impact of technology on society, and the importance of cross-curricular connections, flexibility, and creativity inherent in CS education. Ethical issues and the appropriate use of technology are themes that were not emphasized in the first two Ontario curriculum documents, but both have been a focus in the six curriculum preambles since. The analysis of specific concepts and skills included in the introductory courses shows that some topics such as control structures and the input-storage-processing-output model of the computer have been included in CS curricula for the last 55 years. Other topics and themes, such as program and project design, creativity, expression, the sharing of end products and historical and cultural contexts of CS, while sometimes apparent in the preambles of documents, are noticeably absent in the outcomes and expectations of some of the grade 10 courses.

Considering the growing number of educational jurisdictions beginning to broaden CS concepts and skills in K-12 education, and considering the new CS-related knowledge and skills that students will have developed in elementary school, before entering into secondary CS courses, the results of this study provide an important historical context. The evolution of CS-related courses in Ontario is revealed, encouraging educators, researchers and policy makers to consider historical CS documents to inform effective policy and practice, as well as future research in computing education.

3.1 Introduction

The impetus for research on historical CS curriculum is the increasingly popular trend of integrating and adding CS concepts and skills to mandatory K-12 education around the world. As of 2012, Israel, Russia, South Africa, New Zealand, and Australia had all included CS concepts in their K-12 curriculum (Grover & Pea, 2013, p. 3). In 2016, then President Barack Obama announced the Computer Science for All (CSForAll) initiative, which was intended to expand the scope of CS education for American students in the K-12 grades (Smith, 2016). In 2012, the United Kingdom’s Royal Society published Shut Down or Restart? The Way Forward for Computing in UK Schools which paved the way for the implementation of K-12 computing curriculum that includes a number of CS concepts beginning as early as year 1 (age 5) (The Royal Society, 2017, p. 7).

In Canada, British Columbia has integrated CS concepts into their K-12 Applied Design, Skills and Technologies curriculum (British Columbia Ministry of Education, 2016) while Nova Scotia has developed coding curriculum components to be integrated into a variety of subjects from K-8 (Nova Scotia Department of Education and Early Childhood Development, 2015; Nova Scotia Department of Education and Early Childhood Development, 2016). In Ontario, the Ministry of Education released coding expectations in the 2020 Grades 1-8 Mathematics curriculum (Ontario Ministry of Education, 2020), while in 2021 Alberta released draft CS-related expectations in Grades 1-6 Science curriculum (Alberta Education, 2021)

As CS concepts continue to be included in the public education curriculum of younger grades, it is important to note that the secondary CS curriculum of many educational jurisdictions has a well-established history. In Ontario, Curriculum RP-33 Data Processing (1966) was the first CS-related document released by the Ontario Department of Education (now Ministry of Education). This was followed by seven updates and additions resulting in the most recent Computer Studies curriculum released in 2008 (Ontario, 2008).

What follows is a description of the courses of study of eight Ontario CS curriculum documents, as well as an in-depth analysis of their preambles (introduction, rationale, goals, objectives, etc). Also included, as an additional section for this thesis, is an analysis of specific outcomes and expectations in the four introductory, grade 10 courses that have been implemented in Ontario secondary schools since 1966. This investigation provides findings related to the philosophies, goals, and objectives of secondary CS curriculum in Ontario, as well as a discussion on the potential impact of historic CS documents on recent curriculum reform and CS initiatives.

3.2 Thematic Analysis

This study uses Thematic Analysis (TA) to examine the preambles of historical CS curricula documents. TA offers a systematic way to identify, organize, and offer insight into patterns of meaning or themes within data (Braun & Clarke, 2012). This is appropriate for this study as it sets out to the identify the goals and rationale for a number of historical curricula documents, as well as how the goals and rationale have changed or evolved over the years.

3.2.1 Background

TA was originally developed by Braun and Clarke within the context of Psychology and is now widely used in a number of areas of qualitive research (Braun & Clarke, 2012). TA is useful in identifying patterns across data, and is recognized as being a flexible method of analysis as it can be applied across a variety of theoretical and epistemological frameworks, as well as to a variety of study questions, designs and sample sizes (Kiger & Varpio, 2020).

TA can both describe and interpret data, as it selects and constructs themes through a systematic process of coding data, searching and refining themes, and reporting findings. Rather than examining unique experiences or phenomena, TA is appropriate when searching for common or shared meanings amongst a number of data sets (Kiger &

Varpio, 2020). Within TA, data is not directly coded as themes. Instead, themes are constructed as components of the data are identified, reframed, and connected. While TA is often recognized as being similar to Grounded Theory (GT), which also involves a systematic way to analyze data and generate themes, it’s important to note that TA does not go as far as GT in terms of developing theory (Kiger & Varpio, 2020).

TA approaches can be primarily deductive (top-down) or primarily inductive (bottom-up), but Braun and Clarke (2012) make it clear that all analysis will involve some combination of the two. In a deductive approach, the researcher codes and interprets the data using predefined concepts, ideas, or topics. Alternatively, in an inductive approach the codes and themes come from the contents of the data.

> In reality, coding and analysis often uses a combination of both approaches. It is impossible to be purely inductive, as we always bring something to the data when we analyze it, and we rarely completely ignore the semantic content of the data when we code for a particular theoretical construct - at the very least, we have to know whether it is worth coding the data for that construct. (Braun & Clarke, 2012)

In this study, the data is approached with the intent of identifying the goals and rationale of the curriculum communicated in the documents; however, categories are not predefined. In this way, the study uses a more inductive approach whereby the coding and themes will be constructed from the data. This construction will take place using the six-phase approach described by Braun and Clarke (2021).

3.2.2 Six-Phase Approach

The most widely used method of TA is Braun and Clarke’s (2012) six-phase approach (Kiger & Varpio, 2020). It’s important to note that this approach is not linear, but instead should be iterative, with various phases being revisited throughout the research process. The six-phases are detailed below and include 1) familiarizing yourself with the data, 2) generating initial codes, 3) searching for themes, 4) reviewing potential themes, 5) defining and naming themes, and 6) producing the report.

The first phase involves the researcher familiarizing themselves with the data by reading and re-reading, and by making notes. The process of making notes in this stage is causal rather than systematic, but it is nonetheless important, as Braun and Clarke (2012) point out that this ensures that reading of the data is active, analytical, and potentially critical. By the end of the first-phase, the researcher should be intimately familiar with the data and should be able to begin to identify ideas connected to the research question(s).

After familiarizing themselves with the data, the researcher will now begin generating initial codes during phase two. Braun and Clarke (2012) describe the codes as the individual bricks and tiles of an eventual house that is being built. Data that is relevant to the research question is coded, and can be done using descriptive language, or some interpretation can take place in this initial coding stage.

The third phase involves actively generating or constructing themes from the coded data. Areas of similarity and overlap are identified in the codes, themes are developed, and also connections between the themes begin to be considered. A miscellaneous theme can also be used, to capture any codes that are not clearly connecting with others. This phase concludes with the creation of a thematic map or table outlining potential themes (Braun & Clarke, 2012).

Once the data has been coded and themes have been constructed, phase four involves the recursive process that includes reviewing themes and possibly rearranging codes and collapsing or splitting themes. New themes can be created at this stage, just as some existing themes may be discarded. The phase ends when the themes “capture the most important and relevant elements of the data, and the overall tone of the data, in relation to your research question” (Braun & Clarke, 2012, p. 66).

Phase five involves defining themes, so that it is clear what is distinct and specific about each theme. Braun and Clarke (2012) explain that well established themes will 1) have a singular focus, 2) be related but not overlap, and 3) address the research question. This phase also is the beginning of thematic analysis, which will continue during the sixth phase.

The sixth phase involves producing the report, but should not necessarily begin after the other phases are complete. The writing of various components of the report will most likely occur throughout the process, as notes have been created and a story has begun to emerge (Braun & Clarke, 2012, p. 66).

3.3 Analyzing the Documents

In this study, TA began with the collection of the following eight Ontario secondary curriculum documents (listed in chronological order based on their year of publication):

- Curriculum RP-33, Data Processing (Ontario Department of Education,1966);
- Computer Science – Senior Division (Ontario Department of Education, 1970a);
- Elements of Computer Technology (Ontario Department of Education, 1970b);
- Informatics – Intermediate and Senior Division (Ontario Department of Education, 1972);
- Computer Studies – Intermediate and Senior Division (Ontario Ministry of Education, 1983);
- Computer Studies – Ontario Academic Course (Ontario Ministry of Education, 1987);
- The Ontario Curriculum Grade 11 and 12 - Technological Education (Ontario Ministry of Education, 2000);
- The Ontario Curriculum Grade 10 to 12 – Computer Studies (Ontario Ministry of Education, 2008).

These documents were identified and selected as the most relevant CS-related curriculum by following a trail backwards from the 2008 curriculum document currently in use, as each document lists the documents that it supersedes. As an example, page 2 of the 1983 document indicates “This document supersedes the following guidelines: Computer Science, Senior Division, 1970; Data Processing RP. 33, 1966; Elements of Computer Technology, Senior Division, 1970; Informatics, Intermediate and Senior Divisions, 1972”.

The 2008 Ontario Computer Studies curriculum document was retrieved online at the Ontario Ministry of Education's curriculum website. The remaining seven Ontario curriculum documents were retrieved and scanned from the Ontario Historical Education Collection at the Ontario Institute of Studies in Education library, as these were not available online either through Ontario's Ministry of Education website, or other online repositories. The documents retrieved cover a wide scope of courses. Some, such as the 2008 Computer Studies document, include courses in grade 10, 11 and 12, while others, such as the 1987 Computer Studies – Ontario Academic Course, include only one course.

A preliminary scan of these documents provided information related to the implementation timeframes of each of these documents and the courses of study included in each curricula. These timeframes and courses of study have been described in the findings section.

After a scan of the courses of study was complete, in-depth reading and re-reading of the documents took place, focusing on the preambles of each document, and notes were taken. The preambles from each of the eight documents were then stored in their own digital file to facilitate analysis and coding, which occurred using NVivo software. The following list indicates the components of the preamble that were analyzed for each document, as well as the number of words in each preamble section:

- Curriculum RP-33 - Data Processing (1966): Introduction (182 words) and Foreword (273 words);
- Computer Science – Senior Division (1970): Introduction (536 words) and Scope of the Course (247 words);
- Elements of Computer Technology – Senior Division (1970): Foreword (437 words) and Objectives (199 words);
- Informatics – Intermediate and Senior Division (1972): Introduction (270 words) and Rationale (151 words) and Objectives (96 words);
- Computer Studies – Intermediate and Senior Division (1983): Introduction (276 words), Computers in Daily Life (178 words) and Aims for Computer Studies (327 words);

- Computer Studies – Ontario Academic Course (1987): Rationale (105 words) and Aims (43 words)
- The Ontario Curriculum Grade 11 and 12 - Technological Education (2000): Computer Studies Description (55 words) and Overview (149 words);
- The Ontario Curriculum Grade 10 to 12 – Computer Studies (2008): Importance of Computer Studies in the Curriculum (256 words), Goals of the Computer Studies Program (144 words) and Four Critical Areas of Learning in Computer Studies (71 words).

The coding process involved identifying sentence fragments, and occasionally entire sentences, from the preambles of the eight curriculum documents that revealed information related to the goals and rationale of the curriculum. This stage led to 78 codes, each identified using general education (e.g., student choice and differentiation or how the curriculum was designed and structured) and CS-related terminology (e.g., the use of computer for creative pursuits or the computer as an object of study). The initial codes, as well as the location of references, can be found in Appendix C.

After these codes were identified, 31 themes were constructed by merging and reorganizing codes. The predominant themes, as well as the locations of the coded references, can be found in Appendix D. The process of merging and reorganizing themes continued, with note taking supporting the analysis, and findings were identified and storylines were developed.

3.4 Results

3.4.1 Implementation Timeframes and Courses of Study

Figure 1 indicates the various curriculum documents, and the point in time at which they were superseded by more recent curricula.

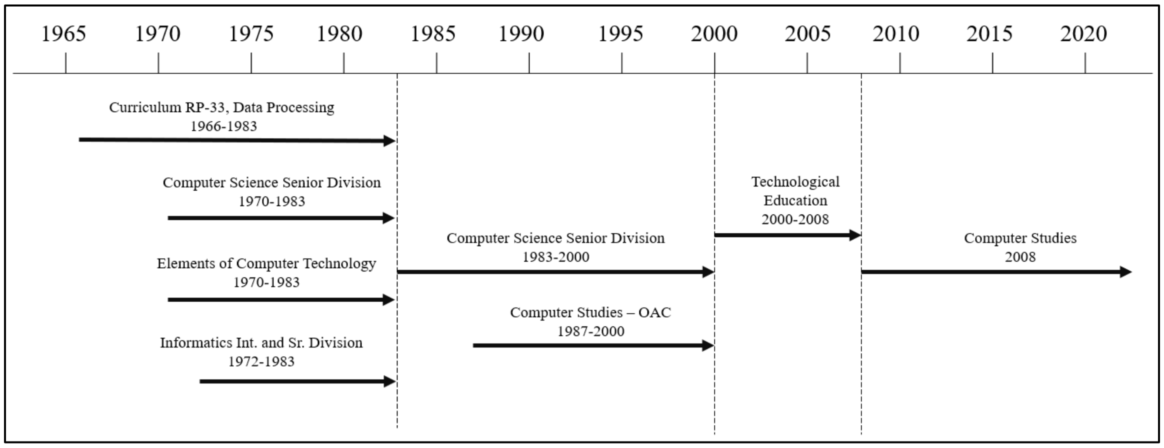

1965
1970
1975
1980
1985
1990
1995
2000
2005
2010
2015
2020
Curriculum RP-33, Data Processing 1966-1983
Computer Science Senior Division 1970-1983
Elements of Computer Technology 1970-1983
Informatics Int. and Sr. Division 1972-1983
Computer Science Senior Division 1983-2000
Computer Studies – OAC 1987-2000
Technological Education 2000-2008
Computer Studies 2008

Figure 1. Computer science related curriculum in Ontario

As indicated in Figure 1, from 1966 to 1983 four different CS-related curricula documents were being implemented at the same time. These included Curriculum RP-33 - Data Processing (1966), Computer Science – Senior Division (1970), Elements of Computer Technology – Senior Division (1970), and Informatics – Intermediate and Senior Division (1972). In 1983, a major restructuring occurred whereby these four documents were replaced by Computer Studies – Intermediate and Senior Division (1983), and in 1987 an additional Ontario Academic Course (OAC) in Computer Studies was added. In 2000, CS-related courses were included in the Technological Education curriculum document and then in 2008, the courses returned to a document titled Computer Studies, which includes courses for grades 10-12. The specific courses of study in each of these documents are listed in Table 3.

Table 3. CS focussed courses of study in Ontario Curriculum (1966-Present)

CS Courses of study 1966-1983	**Curriculum RP-33, Data Processing (1966)** • Principles of Data Processing • Basic Programming • Systems Design • Computer Fundamentals • Business Systems Programming • Unit Record Fundamentals • Business Option Computer Concepts • Business Data Processing • Special Commercial Data Processing	**Elements of Computer Technology (1970)** • Computer Science – Two year course	**Computer Science - Senior Division (1970)** • Elements of Computer Technology I • Elements of Computer Technology II • Elements of Computer Technology III • Computer Applications • Computer Logic • Computer Circuitry	**Informatics – Intermediate and Senior Division (1972)** • Informatics
CS Courses of study 1983-2000	**Computer Studies - Intermediate and Senior Division (1983)** • Grade 10 Introductory Computer Studies (Basic, General, Advanced) • Grade 11 Computer Technology • Grade 12 Computer Technology • Grade 11 Data Processing • Grade 12 Data Processing • Grade 11 Computer Science and Technology • Grade 12 Computer Science • Grade 12 Computer Technology • Grade 11 Data Processing Techniques • Grade 12 Data Processing Systems Analysis and Design		**Computer Studies – Ontario Academic Course (1987)** • Computer Studies	
CS Courses of study 2000-2008	**The Ontario Curriculum Grade 11 and 12 - Technological Education (2000)** • Grade 10 Computer and Information Science • Grade 10 Computer Engineering Technology • Grade 11 Computer and Information Science (C/U preparation) • Grade 12 Computer and Information Science (C/U preparation) • Grade 11 Computer Engineering (College/University preparation) • Grade 11 Computer Engineering (Workplace preparation) • Grade 12 Computer Engineering (College/University preparation) • Grade 12 Computer Engineering (Workplace preparation)			
CS Courses of study 2008-Present	**The Ontario Curriculum Grade 10 to 12 - Computer Studies (2008)** • Grade 10 Introduction to Computer Studies • Grade 11 Introduction to Computer Science • Grade 11 Introduction to Computer Programming • Grade 12 Computer Science • Grade 12 Computer Programming			

3.4.2 Thematic Analysis and Curricula Preambles

The major themes reflecting the goals and rational in preambles of the eight curriculum documents that resulted from Braun and Clarke's (2012) six-phase TA process are summarized in table 4.

Table 4. Appearance of themes in the preambles of CS-related curriculum

	impact of technology on society	automate tasks or solve problems	post-secondary, training, and careers	cross-curricular connections	creativity	differentiation and flexibility based on student needs and interests	ethical issues and appropriate use
Curriculum RP-33 - Data Processing (1966)	✓	✗	✓	✗	✗	✓	✗
Computer Science – Senior Division (1970)	✓	✓	✗	✓	✓	✓	✗
Elements of Computer Technology (1970);	✓	✓	✓	✓	✗	✓	✗
Informatics – Intermediate and Senior Division (1972);	✓	✓	✓	✓	✓	✓	✓
Computer Studies – Intermediate and Senior Division (1983)	✓	✓	✓	✓	✓	✓	✓
Computer Studies – Ontario Academic Course (1987);	✓	✓	✓	✓	✗	✗	✓
The Ontario Curriculum Grade 11 and 12 - Technological Education (2000)	✓	✓	✓	✓	✓	✗	✓
The Ontario Curriculum Grade 10 to 12 – Computer Studies (2008)	✓	✓	✓	✓	✓	✗	✓

All curriculum document preambles indicate the significant impact that technology is having on society and offers this as a rationale for courses to study this phenomenon. The theme of students developing the ability to automate tasks and solve problems with computers was included in all preambles except the initial 1966 Data Processing curriculum. The importance of courses in preparing students for post-secondary and potential careers was also evident, except in 1970. The potential for cross-curricular connections in CS-related courses was evident in all but the 1966 curriculum preamble, while the opportunity for student creativity appeared in all but three preambles. The theme of differentiation and flexibility in course design and delivery was included in the preambles of early documents, but in 1987 and afterwards, references to these themes were not included. This is not to say that the courses did not embody these themes, but just that these themes did not emerge in the initial preambles. Finally, issues surrounding the ethical use of computers and related technologies did not appear in early preambles but remained in documents after the mention first appeared in 1972.

3.5 Discussion

3.5.1 Curriculum Documents and Courses of Study

The location of CS-related courses in Ontario's secondary curriculum is interesting when analysed historically. The 1966 Curriculum RP-33, Data Processing document refers to itself as a Business and Commerce document, and so it's significant that it is within the discipline of Business that CS concepts and skills first make their appearance in secondary education in Ontario. The courses in this document include Basic Programming and Computer Fundamentals, but even the more business focused courses include what would be known as specific CS topics. As an example, the Principles of Data Processing Grade 10 course includes the study of binary, octal and hexadecimal number systems, topics that connect to important mathematics learning as well as CS concepts.

After the 1966 document was released, three other documents followed (Computer Science, Elements of Computer Technology, and Informatics), indicating that there must have been a need to expand the offering of CS-related courses, beyond the Business and Commerce document. The courses of studies in these three curricula were being implemented in Ontario at the same time as the courses of study in the 1966 document, which means that a total of 17 CS-related courses were made available for implementation (see Table 3). The major reorganization, in 1983, of all CS-related course within the Computer Studies document is significant. It meant that CS-related courses now had a single and formal home and would be organized under the heading of Computer Studies. Today, the secondary CS-related courses still find themselves within a curriculum document entitled Computer Studies; however, from 2000-2008 these were found within the Technological Education document.

This historical analysis of Ontario CS-related curriculum begs the question, where do CS-related courses belong? Are CS concepts and skills fundamental to business or to technological education, or do the CS-related courses need their own Computer Studies curriculum document? A jurisdictional scan of current CS-related secondary courses across Canada adds to this complexity. In British Columbia, grade 11 and 12 courses such as Computer Information Systems and Computer Programming are found within the Applied Design, Skills, and Technologies curriculum while grade 11 and grade 12 CS courses are offered within the Mathematics curriculum (British Columbia Ministry of Education, 2018a; British Columbia Ministry of Education, 2018b; British Columbia Ministry of Education, 2018c; British Columbia Ministry of Education, 2018d; British Columbia Ministry of Education, 2018e; British Columbia Ministry of Education, 2018f). In Saskatchewan, grade 11 and grade 12 Computer Science courses are offered from within the Science curriculum, providing an additional alternative (Saskatchewan Ministry of Education, 2018a; Saskatchewan Ministry of Education, 2018b). This leaves five different contemporary or historical homes for CS-related courses in Canadian jurisdictions that include Business, Computer Studies, Mathematics, Science, and Technological Education.

As the broadening of CS education continues, policy makers are left with a wide variety of precedent setting options for the organization of CS-related courses. As an alternative, some might even begin to conclude that if CS-related concepts and skills have applications in all of these disciplines, then is it possible to integrate these courses, concepts or skills within the different subjects and contexts, rather than in a separate, isolated, CS-related document? If this question was explored, could it potentially lead to the end of formal CS-related curriculum, as instead the concepts and skills would be redistributed to other areas? Or is it possible that a redefinition of CS curriculum takes place, one that involves a very narrow focus on the computer itself, while other subject areas integrate relevant CS-related concepts and skills within their specific disciplines and contexts? This research raises these questions as important for consideration as the broadening of CS continues. It also raises these questions for other subject areas, such as those included in the STEM (Science, Technology, Engineering and Mathematics) related subjects, where there are potential advantages to a cross-curricular or cross-disciplinary approach. How can these subject areas be organized in formal curriculum documents, if the concepts and skills overlap extensively, or are applicable to several subject areas?

3.5.2 Focus of Preambles in CS-Related Curricula

Thematic Analysis revealed eight important goals and rationale related to the CS-related curricula documents. These included:

- the impact of technology on society;
- automating tasks;
- solving problems;
- post-secondary and career preparation;
- cross-curricular connections;
- creativity;
- differentiation and flexibility based on student needs and interests; and
- ethical issues and appropriate use.

3.5.2.1 Impact of Technology, Automating Tasks, and Solving Problems

It is no surprise that the impact of technology on society was discussed in all curriculum documents, but what might be surprising is just how forward thinking the 1966 and 1970 documents were in this regard. The preamble of the 1966 document included six references to the impact of technology on society. These references included a wide range of areas impacted by technology including business, industrial, and government organizations as well as other social institutions, management and economic production and distribution patterns. The introduction and foreword of the 1966 document also predicts rapid improvement in technology and more sophisticated methods of processing data and acknowledged the importance of leveraging the resultant knowledge: “The more effectively we use these new tools to produce and store information, and the more skillfully we use the resultant knowledge, the greater the benefit to all society" (Ontario Department of Education, 1966, p. iii).

The preamble of the 1970 document also included references to the impact of technology on society and also appears quite forward thinking. The introduction of the document begins with the following: "The influence of technology on our society is increasing rapidly" (Ontario Department of Education, 1970a, p. 3) indicating that this area is a major impetus for the development of these CS courses. The 1970 document also highlights the relationship between humans and computers and how the computer is allowing for human mental effort to be extended. This connects to the second major theme that was introduced in all documents, which was the recognition and importance of the computer as a tool to automate tasks and solve problems. All documents except the initial 1966 documents make reference to these themes, and they are often referred to in contemporary arguments for why students should learn to program a computer.

3.5.2.2 Post-Secondary and Career Preparation

Post-secondary and career preparation was included in all document preambles, which speaks to the popular economic argument for the broadening of CS education. Passey (2017) describes the economic argument as workforce centered, 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 CS-related concepts and skills will be valuable for future careers. This argument, included in all eight curricula, continues to be used by governments in their broadening of CS mandates. Canada's CanCode initiative includes the economic argument as the main principle of the program:

> CanCode aims to equip Canadian youth, including traditionally underrepresented groups, with the skills they need to be prepared for further studies, including advanced digital skills and science, technology, engineering and math (STEM) courses, leading to the jobs of the future. Canada's success in the digital economy depends on leveraging our diverse talent and providing opportunity for all to

> participate - investing in digital skills development will help to achieve this. (Government of Canada, 2019)

Considering that these curriculum documents cover the secondary grades, it is not surprising that the economic argument is included in curriculum preambles. What will be interesting to see; however, is whether the economic argument becomes integrated into potential CS-related curricula in the K-8 grades. This phenomenon has begun to occur in educational jurisdictions in Canada, but there has not been research uncovering whether or not the economic argument is outlined as a major impetus for this change to curriculum.

3.5.2.3 Cross-curricular Connections, Creativity, Differentiation and Flexibility

The cross-curricular nature of CS-related concepts and skills, as well as their potential for creativity and the need for differentiation and flexibility in courses were all interesting themes to uncover in the curriculum documents. The nature of secondary curriculum and implementation is often siloed and isolated into specific disciplines, making cross-curricular connections difficult. In addition, there is often a need for standardized and aligned expectations and outcomes across a jurisdiction, which results in themes such as creativity, differentiation, and flexibility being omitted. The fact that all of these themes appear in many of the historical documents is perhaps a result of the inherent nature of CS-related instruction and pedagogy. Over 50 years ago, the preamble of the 1970 Computer Science document captured these associated themes well:

> The boundaries of the material are virtually limitless, largely because of its emphasis on problem-solving. Thus, the student has an opportunity to pursue problems in any subject area and to study the solutions to such problems to any depth he desires. The flexibility inherent in the suggested approach permits individual research projects, class research projects, and any other challenging venture that either students or teachers may initiate.

> The emphasis on problem-solving permits another flexibility: that is, the level at which the course is studied. The number of levels is almost infinite, largely because there is such a diversity of difficulty in the problems that can be solved through the use of a computer. (Ontario Department of Education, 1970a, p. 4)

This excerpt also connects directly to theory in the CS education literature. Seymour Papert, described by Stager (1996) as the father of computing education, discussed the idea of coding as a low floor, high ceiling context (Papert, 1993), whereby students can enter into the learning from a simple entry point (low floor), but can extend projects with almost unlimited depth and sophistication (high ceilings). Gadanidis et al. (2017) describe how the low floor, high ceiling concept connects to the themes of cross-curricular connections, creativity, differentiation, and flexibility:

> Coding in a low-floor and a high-ceiling environment also supports student agency and gives students ownership of their learning. Students writing code to model a pattern or a relationship are in control. There are many different ways to solve a problem with code and students can use methods that personally make sense. They can also deviate from the task to investigate related problems. (p. 3)

By investigating related problems, it is possible for students to make cross-curricular connections, or for educators to point these out, while the different ways to solve problems supports the expression and development of creativity. Meanwhile, providing activities that facilitate both basic entry points (low floor) and the potential for added depth and sophistication (high ceiling), activities are differentiated for students who come to class with varying skills, knowledge and experiences in CS. As new and exciting CS-related curriculum is developed in secondary, and potentially in the K-8 grades, it will be interesting to see how educational jurisdictions integrate cross-curricular connections, creativity, differentiation and flexibility in CS curriculum design. On the one hand, a curriculum is often siloed and isolated in its own document and in its implementation, yet historical curriculum and CS-related theory highlights the need for cross-curricular connections. Additionally, curriculum making is often about standardizing expectations and outcomes across a jurisdiction, yet historical curriculum and theory point to a need for creativity on behalf of the students, and flexibility and differentiation on behalf of the teacher. It is expected that these opposing forces will be important for policy makers to consider and resolve as curriculum potentially expands beyond the isolated, secondary CS-related curriculum documents.

3.5.2.4 Ethics and the Appropriate Use of Technology

Ethics as well as the appropriate use and the impacts of technology are becoming important topics within recent CS curriculum reform, and it is encouraging to note that these issues are well discussed in the preambles of CS curricula since 1972. The K-12 Computer Science Framework (K-12 Computer Science Framework Steering Committee, 2016), led by the Association for Computing Machinery, Code.org, Computer Science Teachers Association, Cyber Innovation Center, and National Math and Science Initiative includes the impacts of computing as one of five main, core concepts. This core concept includes culture, social interactions, safety, law, and ethics as subcomponents. In Canada, Canada Learning Code’s Pan-Canadian K-12 CS framework (Canada Learning Code, 2020) includes Technology and Society as one of five main focus areas. This focus area also includes ethics, safety, and the law as one of six focus areas.

As jurisdictions update their secondary CS-related curricula or begin to integrate CS concepts into other subjects and grades, ethical issues and the appropriate use of technology will need to be an important component. This is especially true with the emergence of new technologies, such as Artificial Intelligence (AI), that may be making their way into CS curricula. The AI4K12 initiative, sponsored by the Advancement of Artificial Intelligence and the Computer Science Teachers Association, identified five big ideas of AI for students to learn, one of which includes the impact of AI on society (Touretzky et al., 2019). This emphasizes the notion that while there is a need to teach the concepts and skills required to understand, apply and leverage AI, Machine Learning (ML), and associated concepts, there will also need to be room for the associated ethical components. As jurisdictions consider the integration of ethics and appropriate use of technology within the curriculum, age-appropriate expectations and outcomes, learning progressions, and resources related to these important areas will need to be considered. It should be noted that this work has begun by such groups as AI4K12, and researchers such as Gadanidis and Hughes who have begun developing AI-related stories for young students, such as AI Farm (Gadanidis & Hughes, 2019).

3.5.2.5 Preambles and Gender in CS Education

During the analysis of the preambles of historical CS curriculum in Ontario, it was evident that the language used when referring to students had changed throughout the years. In the 1970 Computer Science document, the language used indicates that there is an underlying assumption that male students will be enrolled in the course:

> The student of Computer Science should acquire this basic understanding of the computer, and he should also learn how to make it work for him. In achieving these objectives, he should not only learn of the tremendous power of and capabilities of the computer but also of its limitations and its dependence on human intelligence. He should appreciate that the computer extends the human brain just as machines have extended human muscle power since the time of the industrial revolution. (Ontario Department of Education, 1970b, p. 3)

This type of language continues throughout the preamble of the 1970 document, with "he" and "his" pronouns appearing nine times, while "she" or "her" do not appear at all. This is interesting, considering that within the same preamble, the document acknowledges CS courses being open to all students: "The choice of enrolling in a Computer Science class should, ideally be open to any students indicating an interest in and enthusiasm for the study of computers and computing" (p. 4). It is also interesting considering the importance of equity, diversity, and inclusivity in work being done to broaden participation in CS education, specifically surrounding the under representation of female students in CS fields and courses.

In 2002, Jane Margolis and Allan Fisher published their influential book *Unlocking the clubhouse: Women in computing*, which presented computer education as a clubhouse for boys that was resulting in women and girls being left out of CS. The authors discovered a number of influences contributing to a gender gap in computing education, and they referred to these influences as the doors, walls and windows of the computing clubhouse. The use of only “he” in the preamble of the 1970 document is perhaps a very good example of these doors, walls and windows, that fail to support female students accessing or remaining in CS courses.

In order to further explore this direction of inquiry, the preambles and additional components of the other curriculum documents were scanned, with the goal of identifying the inclusion of different pronouns in the texts. This scan revealed the following:

- The preamble of the 1970 Elements of Computer Technology document did not use gender specific pronouns when referring to students; however, a brief analysis of the course descriptions included in the document indicated that when referring to students, “he” and “his” was used 10 times while “she” and “her” were not used at all.
- The preamble of the 1972 Informatics document used “his” once, within the objectives section, when referring to students, while “she” or “her” was never used. In addition, within the Developing a Local Course section of the document, the pronouns “he” and “his” were often used when referring to the teacher of the course (“he” appeared five times, while “his” appeared six). The following is an example:
  In developing a course, the teacher must weigh several factors. Naturally he will be strongly influenced by his own strengths and interests, but he should also be mindful of the interests and preferences of his students. (Ontario Department of Education, 1972, p. 7).
- In 1983, the pronoun “their” is used when referring to students, with no appearance of “he”, “his”, “she” or “her”. This is true for all of the remaining CS-related curricula documents that appear after 1983.

3.6 Implications and Future Studies

The results of this study on curriculum preambles are important when situated within current educational policy and curriculum reform. A number of educational jurisdictions are beginning to broaden participation in CS in K-12 education, but there is often a feeling that CS in K-12 education is brand new, with little precedent. Acknowledging that in Ontario, CS-related curriculum dates as far back as 1966, and that many of the important themes in the broadening of CS discourse have been recognized in historical documents, is important. Failure to acknowledge and learn from both the positive and negative aspects of these documents would mean ignoring resources that can be used to inform important policy-making practice and research.

In addition, the exploratory nature of this study raised interesting questions that can serve as seeds of future studies that focus on historical CS curriculum to inform contemporary curriculum. These include future investigations surrounding the connections between CS concepts and skills and other subject areas. In Ontario, historical documents have acknowledged the importance of cross curricular connections in CS curricula and the CS courses themselves have been placed in Business, Computer Studies, Computer Science/studies, Informatics, and Technological Education, while elsewhere we see secondary CS courses being placed in Mathematics and Science curriculum documents. What will the future placement of CS-related courses be? Is CS such an interdisciplinary subject that there is the possibility for integrating CS courses in numerous or all subject areas? Or is there a need to expand the secondary CS curriculum, by integrating the various subject areas and disciplines into the CS courses themselves?

Finally, the identification of only the “he” and “his” pronouns appearing in early curriculum documents, when referring to CS students and teachers, has important implications when considering issues of underrepresentation in CS education. What was the impact of the use of this language on enrolment in CS courses, and in the representation of teachers? Are there more examples of this exclusionary language in historical, or contemporary curriculum or resources and if so, what is the impact?

3.7 Additional Dissertation Section – Grade 10 Curriculum Components

Chapter 3 was written with the goal of focusing specifically on the preambles of historical curriculum documents in order to investigate the general approaches that have been identified in CS education at the secondary level. In order to add depth and understanding within the context of this larger dissertation, this additional section was added, which looks specifically at the concepts and skills included in the historical, grade 10 CS courses in Ontario, as well as some of the pedagogical approaches represented in the documents. The grade 10 course was selected for analysis as it is an introductory course that teaches foundational CS concepts and skills. Introductory, secondary courses will be greatly influenced by new CS concepts and skills that are being introduced in the K-8 grades in a number of jurisdictions, and it is therefore an important course for analysis and understanding.

3.7.1 Courses and Main Concepts

The initial analysis of the eight historical CS-related curriculum documents reveals that there are four grade 10 courses implemented from 1966 to present day. These grade 10 courses represent the first introductory CS-related course available to students, as none of the curriculum documents include a grade 9 course. As a result of these being from the same jurisdictions (Ontario), and building upon each other, they represent an interesting evolution of introductory CS courses.

Table 5, lists the names of the four courses from the curriculum documents, as well as the main topics that are identified. The main topics do not appear in this order in the documents, they have been reorganized in order to show the progression of each broad category over the years. The main topics are those that are highlighted as either major unit or concept headings, or are identified in overall expectations or more recent documents.

Table 5. Grade 10 introductory CS-related courses, from 1966 to present day

	1966 - Principles Of Data Processing Grade 10	1983 - Introductory Computer Studies Grade 10	2000 - Computer and Information Science Grade 10	2008 - Introduction to Computer Studies Grade 10
Technology and society	Man and His Environment Important Uses of the Computer in Today's Society	Impact of microelectronic technology on society	Impact of computers and technologies	Social impact Environmental stewardship and sustainability Ethical issues
The computer and its operation	How the Computer Works in its Simplest Form Introduction to Electronic Computers Storage Devices: advantages and disadvantages Output Media and Data Presentation Central Processor Computer Input Media	Operating the computer Computer electronics	Hardware, interfaces, and networking	Hardware components Software products Operating systems Home Computer Networking Maintenance and Security
Programming	Data Processing Coding Manual and Mechanical Methods Electro-mechanical Methods	Programming the computer Information Processing	Programming concepts	Programming concepts Writing programs Code maintenance
Careers and post-secondary	Forecast of vocational opportunities in the data processing field	Computer-related careers	Related careers	Postsecondary Opportunities
Historical context	Development of Devices to Improve Information Processing		Evolution of programming languages	
Algorithms & representation	Number Systems Manipulation	Computer Science	Logic	
Problems and Design	Summary, Review and Application		Problem solving and design	

As indicated in Table 5, the main topics of technology and society, the computer and its operation, programming, and career and post-secondary opportunities were included in all four courses, from 1966 to present.

Topics surrounding the historical development of CS-related topics and technologies were only present in the 1966 and 2000 documents, meaning that students did not learn about the historical contexts or the evolution of technologies when enrolled in the 1983 course, nor do they learn about them presently, with the 2008 course. In the 2000 document, the extent of historical context in topics only includes students investigating the evolution of computer programming languages. In 1966; however, a much more in-depth exploration of historical context is included, as students were expected to learn about all of the following:

- evolution of the computer;
- development of the accumulation and transfer of knowledge;
- development of methods of assembling, writing, and recording information (records on stone, clay tablets, paper, disk, tape);
- development of systems to express specialized information (weights, measures, money, maps, accounting);
- development of automation (sail, pump, gear, lever, wheel, thermostat, conveyor belt, elevator, computer);
- development of devices to improve information processing (Abacus, Napier's bones, Pascal's gears, slide rule, manual calculators, electro-mechanical calculators, postage meters);
- history of electro-mechanical methods (Hollerith, Powers); and
- early storage devices (electrostatic, delay lines, electronic tubes).

Program and project design, like historical context, is only present in the 1966 and 2000 courses, but not included in the 1983 or 2008 courses. In terms of topics surrounding algorithms and representation of data, the most recent 2008 grade 10 course does not include this topic, while all three previous courses did.

3.7.2 Other Concepts

As the investigation of main topics took place (those that were unit headings or that were included as overall expectations in the documents), it was clear that other, minor topics were either apparent, or noticeably absent in some or all of the four documents.

The concepts of control structures in CS, which include such things as the sequencing and repetition of instructions, as well as conditional statements (decisions), were included in all grade 10 courses. 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). The model of the computer as an input, storage, processing and output device was also included in all four courses. An example of how it was included in the 1966 documents is as follows: "Block diagram of computer: explain in terms of input, central processing, output and auxiliary storage. Trace typical path of information processing through a computer using the block diagram as sectionalized above" (Ontario Department of Education, 1966, p. 3). The inclusion of these two topics, in all four courses since 1966, indicates that they are fundamental concepts required in introductory CS courses. Like some of the main topics listed above, leaving these topics out of new, future courses would require some careful thought and strong rationale.

Noticeably absent in all four of the CS courses were concepts that are apparent in some of the contemporary discourse around the broadening of CS, including creativity, expression, and the sharing of end products with others. Both Resnick (2017) and Kafai (2016) write extensively about creativity, expression and the social aspects of computation in their models of Computational Fluency and Computational Participation respectively. None of the courses included outcomes or expectations that involved students being creative or finding ways to express themselves and share projects with others. Instead, the content appeared more technically based, focused on specific CS skills and concepts.

3.7.3 Pedagogical Approaches

An analysis of the pedagogical approaches represented in the four grade 10 courses was done by identifying and comparing the verbs used in the curriculum outcomes and expectations. This analysis provides insight into what it is expected of students in terms of how they engage with the content and skills listed in the course. Interestingly, the earlier curriculum documents, from 1966 and 1983, appear to rely less heavily on verbs to explain what students are expected to be doing in the course. The 1966 document reads much more like a list of concepts, skills, or facts, but there is little guidance in terms of how students are to engage with these components. The 1983 document includes the use of specific verbs within the Operating a Computer and Programming a Computer sections, but the other sections of the document are written much like the 1966 document, indicating the concepts that “students should gain an understanding” of.

In contrast, all of the 45 specific expectations in the 2000 course, and all of the 48 specific expectations in the 2008 course, begin with a verb. This provides important insight for the teacher, in terms of how the students will engage with the material. Table 6 shows the number of times that different verbs were used in the 2000 and 2008 grade 10, introductory course (note that some expectations include more than one verb):

Table 6. Occurrence of verbs in the 2000 and 2008 grade 10, introductory CS courses

Verb	2000 Computer and Information Science Grade 10	2008 Introduction to Computer Studies Grade 10
describe	8	19
use	10	8
identify	3	6
explain	5	6
write	5	5
research, demonstrate, assess, understand	0	2
correct	1	1
contrast, compare	2	1
plan, determine	0	1
state, define	2	0
comply, find, design, solve, verify, develop, maintain, incorporate, trace, validate	1	0

It is interesting to note that the verb “describe” appeared 19 times in the current, introductory CS course, more than twice as often as any other verb. While this 2008 document does not include any explicit mention of communication skills, it is clear that communication will be important, as almost half of the curriculum expectations include students “describing” concepts and/or skills. This observation has important considerations for both classroom pedagogy and assessment as a focus on communicating concepts and skills, rather than on demonstrating them, could potentially have an impact on:

- The time spent by students planning, writing, executing and debugging programs versus that time spent describing and communicating concepts;
- The potential hands-on, exploratory nature of CS learning and the constructionist learning theory (Papert, 1993) often at the heart of CS education;
- The number of students who chose to enrol in the course and the engagement of students once enrolled.

3.8 Conclusion

As this thesis moves on to explore enrolment, equity, diversity, and inclusivity in Chapter 4, and contemporary curriculum approaches in the K-8 grades in Chapter 5, the findings from this historical analysis of secondary CS curriculum provide important context. Current CS-related curriculum in Ontario is found in the 2008 secondary Computer Studies document, and it will be interesting to explore both the student enrolment in these courses, and whether or not female and male students are equally represented. Historically, CS-related courses were also included in Business and Technological Education curriculum documents, while in other jurisdictions CS courses are offered within Mathematics and Science. These connections to other disciplines will be interesting to consider as an analysis of the different approaches to CS-related curricula in the K-8 grades is explored. In addition, this chapter revealed that curriculum content related to technology and society, the computer and its operation, programming, and related careers and post-secondary opportunities are well entrenched in Ontario CS curricula, appearing in all courses since 1966. Other areas of focus such as creativity, expression and the sharing of projects with others have been a part of preambles and document introductions, but have not always been included in the lists of concepts and skills to be taught. Whether or not the inclusion and exclusion of these topics is relevant in terms of enrolment, equity, diversity, and inclusivity and whether or not these topics appear in novel K-8 curriculum will be further explored.

3.9 Chapter 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

Blumer, H. (1969). *Symbolic interactionism: Perspective and method*. University of California Press.

Braun, V., & Clarke, V. (2012). Thematic analysis. In H. Cooper, P. M. Camic, D. L. Long, A. T. Panter, D. Rindskopf, & K. J. Sher (Eds.) *APA Handbook of Research Methods in Psychology* (pp. 57-71). American Psychology Association. https://doi.org/10.1037/13620-004

British Columbia Ministry of Education. (2016). *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. (2018). 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. (2018a). Applied design, skills and technologies: Computer information systems grade 11. https://curriculum.gov.bc.ca/sites/curriculum.gov.bc.ca/files/curriculum/adst/en_adst_11_computer-information-systems_elab.pdf

British Columbia Ministry of Education. (2018b). Applied design, skills and technologies: Computer programming grade 11. https://curriculum.gov.bc.ca/sites/curriculum.gov.bc.ca/files/curriculum/adst/en_adst_11_computer-programming_elab.pdf

British Columbia Ministry of Education. (2018c). Applied design, skills and technologies: Computer information systems grade 12. https://curriculum.gov.bc.ca/sites/curriculum.gov.bc.ca/files/curriculum/adst/en_adst_12_computer-information-systems_elab.pdf

British Columbia Ministry of Education. (2018d). Applied design, skills and technologies: Computer programming grade 12. https://curriculum.gov.bc.ca/sites/curriculum.gov.bc.ca/files/curriculum/adst/en_adst_12_computer-programming_elab.pdf

British Columbia Ministry of Education. (2018e). Mathematics: Computer science grade 11.

https://curriculum.gov.bc.ca/sites/curriculum.gov.bc.ca/files/curriculum/mathematics/en_mathematics_11_computer-science_elab.pdf

British Columbia Ministry of Education. (2018f). Mathematics: Computer science grade 12. https://curriculum.gov.bc.ca/sites/curriculum.gov.bc.ca/files/curriculum/mathematics/en_mathematics_12_computer-science_elab.pdf

Canada Learning Code. (2020*). Learning for the digital world: A pan-Canadian K-12 computer science education framework*. https://k12csframework.ca/wp-content/uploads/Learning-for-the-Digital-Future_Framework_Final.pdf

Cherryholmes, C.H. (1992). Notes on pragmatism and scientific realism. *Educational Researcher, 21(6)*, 13-17. https://doi.org/10.3102/0013189X021006013

Cohen, L., Manion, L., & Morrison, K. (2018). *Research methods in education* (eighth edition). Abingdon, Oxon.

Creswell, J. W. (2012). *Qualitative inquiry and research design: Choosing among five approaches*. Sage Publications.

Gadanidis, G., Brodie, I., Minniti, L., & Silver, B. (2017). Computer coding in the K-8 mathematics curriculum? *What works: Research into practice*. http://www.edu.gov.on.ca/eng/literacynumeracy/inspire/research/Computer_Coding_K8_en.pdf

Gadanidis, G., & Hughes, J. M. (2019). AnImal farm. LearnX.

Government of Canada. (2019). *CanCode*. https://www.ic.gc.ca/eic/site/121.nsf/eng/home

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.org10.3102/0013189X12463051

Herman, N. J., & Reynolds, L. T. (1994). *Symbolic interaction: An introduction to social psychology*. General Hall.

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

Kiger, M. E., & Varpio, L. (2020). Thematic analysis of qualitative data: AMEE Guide No. 131. *Medical teacher*, 42(8), 846-854. https://doi.org/10.1080/0142159X.2020.1755030

K-12 Computer Science Framework Steering Committee. (2016). *K-12 Computer Science Framework*. https://k12cs.org/

LaRossa, R., & Reitzes, D. C. (1993). Symbolic interactionism and family studies. In P. G. Boss, W. J. Doherty, R. LaRossa, W. R. Schumm, & S. K. Steinmetz (Eds.*), Sourcebook of family theories and methods: A contextual approach* (pp. 135-163). Springer Science. https://doi.org/10.1007/978-0-387-85764-0_6

Margolis, J., & Fisher, A. (2002). *Unlocking the clubhouse: Women in computing*. MIT press.

Musolf, G. R. (2009). The essentials of symbolic interactionism: A paper in honor of Bernard N. Meltzer. *Studies in symbolic interactionism*. Emerald Group. https://doi.org/0.1108/S0163-2396(2009)0000033021

Nova Scotia Department of Education and Early Childhood Development. (2015). *Information and communication technology – Essential learning outcomes 2015-2016*. https://www.ednet.ns.ca/files/curriculum/ITC-P-3ProgressionChart-RevAug26-2015.pdf

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

Ontario Department of Education. (1966). *Curriculum RP-33: Data processing*.

Ontario Department of Education. (1970a). *Computer science: Senior division*.

Ontario Department of Education. (1970b). *Elements of computer technology*.

Ontario Department of Education. (1972). *Informatics: Intermediate and senior division*.

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

Ontario Ministry of Education. (1987). *Computer studies: Ontario academic course.*

Ontario Ministry of Education. (2000*). The Ontario curriculum grade 11 to 12: Technological education*.

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

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

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

The Royal Society. (2012). *Shutdown or restart? The way forward for computing in UK schools*. https://royalsociety.org/-/media/education/computing-in-schools/2012-01-12-computing-in-schools.pdf

The Royal Society. (2017). *After the reboot: Computer education in schools*. https://royalsociety.org/~/media/policy/projects/computing-education/computing-education-report.pdf

Saskatchewan Ministry of Education. (2018a). Computer Science 20. https://www.edonline.sk.ca/webapps/moe-curriculum-BB5f208b6da4613/CurriculumHome?id=446

Saskatchewan Ministry of Education. (2018b). Computer Science 30. https://www.edonline.sk.ca/webapps/moe-curriculum-BB5f208b6da4613/CurriculumHome?id=444

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

Stager, G. S. (2016). Seymour Papert (1928-2016). *Nature*, *537*(7620), 308-308. https://doi.org/10.1038/537308a

Touretzky, D., Gardner-McCune, C., Martin, F., & Seehorn, D. (2019). Envisioning AI for K-12: What should every child know about AI? In *Proceedings of the AAAI Conference on Artificial Intelligence*, *33*(1), 9795-9799. https://doi.org/10.1609/aaai.v33i01.33019795

Chapter 4

4 Enrolment and Underrepresented Groups in Computer Science Education

Current literature surrounding the broadening of CS concepts and skills indicates that there are a number of underrepresented groups in CS K-12 Education. This chapter focusses on the underrepresentation of female students in Ontario secondary CS courses. The chapter reveals a significant gender gap in these courses, and also finds that overall enrolment is lowest in the grade 11 and grade 12 College pathway courses. Considering recent research (Pichette et al., 2020) and government initiatives (Alphonso, 2021) related to de-streaming unequitable course designations and pathways (such as Academic and Applied), it is felt that findings from this chapter related to the low enrolment in College pathways can provide a starting point upon which to further research the potential underrepresentation of other groups within these courses.

While Chapter 3 explored the traditional, optional secondary course implementation of CS in the K-12 grades, this chapter builds upon this work by seeking to understand how enrolment patterns in these courses have changed over time. The chapter explores overall student enrolment in Ontario's secondary CS courses, from 2011-2018, as well as the important theme of equity, diversity, and inclusivity in CS by examining the enrolment of female and male students. This theme is not only a major component of recent initiatives meant to broaden CS education, including Canada's CanCode (Government of Canada, 2019c) and the US's CS For All initiative (Smith, 2016), it is also often discussed in K-12 CS education literature.

Almost twenty years ago, Margolis and Fisher (2002) presented computer education as a clubhouse for boys where women and girls are left out of computer science (CS). While the authors acknowledged that "women are surfing the web in equal proportion to men, and women make up the majority of Internet consumers" (p. 2), women were not learning to invent, create and design with computer technology, a concern that lead to missed educational and economic opportunities. Through interviews, classroom observations, conversations with faculty, and analysis of relevant data, the authors discovered a number of influences contributing to a gender gap in computing education. They called these influences the doors, walls and windows of the computing clubhouse.

Now, as computing technology becomes ubiquitous, as the expansion of CS education takes place in a number of K-12 educational jurisdictions, and as the economic opportunities resulting from computing education have expanded, it is important to develop local and current perspectives on the issue.

4.1 Research Rationale

The impetus for this research centers on the fact that CS is becoming a nascent focus of curriculum initiatives in Ontario, Canada and abroad. In Canada, the Ontario Ministry of Education recently announced a strategy to revise secondary school CS curricula in an effort to focus on developing job skills such as computational thinking and coding (Ontario Ministry of Education, 2019), while Canada's federal government announced an additional $80 million of funding to their CanCode coding initiative (Department of Finance Canada, 2021). In the K-8, grades British Columbia (British Columbia Ministry of Education, 2016), Alberta (Alberta Education, 2021), Ontario (Ontario Ministry of Education, 2020), New Brunswick (New Brunswick Department of Education and Early Childhood Development, 2016) and Nova Scotia (Nova Scotia Department of Education and Early Childhood Development, 2016) all include coding and CS concepts in their current or draft K-8 curriculum while beyond Canada the integration of coding into K-8 education has become an international phenomenon (Gadanidis et al., 2017). Research seeking to provide insight into enrolment in current CS courses, as well any existing gender gaps in CS education, is critical considering the number of initiatives that have been implemented to broaden CS education, and considering the potential impact and missed opportunities that result from a CS student population and workforce lacking in diversity.

4.2 The Broadening of CS Education

In recent years, increased attention has been given to the broadening of CS concepts and skills in K-12 education. A number of theoretical approaches related to the broadening of

CS in K-12 education were presented in Chapter 2 of this thesis and are explored further, in the specific context of K-8 curricula, in Chapter 5. While much of this theory has its foundation in Papert's (1993) work, it has also been recognized that Wing's (2006) Computational Thinking has been influential in broadening CS initiatives and in catching the attention of the CS education community (Grover & Pea, 2013).

4.2.1 Ontario and Canada

In Ontario, the secondary Computer Studies curriculum, which was analyzed within its historical context in Chapter 3, is currently undergoing revisions (Ontario Ministry of Education, 2019) while in British Columbia secondary CS courses were revised in 2018 and situated within the Mathematics curricula (British Columbia Ministry of Education, 2018a; British Columbia Ministry of Education, 2018b). Likewise, in Saskatchewan, secondary CS courses were revised in 2018; however, unlike British Columbia, these courses are now included in the Science curriculum (Saskatchewan Ministry of Education, 2018; Saskatchewan Ministry of Education, 2018b). In addition to curriculum revisions in the secondary grades, new K-8 curriculum from a variety of provinces now includes coding and computational thinking concepts and skills, the details of which will be explored further in Chapter 5. While these curriculum updates and revisions have been led by provincial Ministries of Education, there has also been a large federal initiative in Canada, where money to broaden CS education was provided to non-profit organizations.

The CanCode initiative began in 2017 with an initial commitment, from the Canadian federal government of $50 million (Department of Finance Canada, 2017). In 2019 and 2021, the federal budgets earmarked an additional $60 million (Department of Finance Canada, 2019) and $80 million (Department of Finance Canada, 2021) respectively for the program, resulting in provided or promised funding for the programming totaling $190 million. The CanCode program was developed to help provide coding and digital skills education to more young Canadians (Government of Canada, 2019c), and is listed as part of an action item related to Canada's Digital Charter: Trust in a Digital World (Government of Canada, 2021). In its first two years, the program had provided more than 800,000 K-12 students and 40,000 teachers with opportunities to learn coding and digital skills (Government of Canada, 2019a). These figures included 350,000 girls, over 68,000 Indigenous students, over 100,000 youth at risk, and 34,000 newcomers to Canada (Government of Canada, 2019a).

One example of a non-profit organization that was provided with CanCode funding is Canada Learning Code (CLC), whose vision is “a prosperous Canada in which all people have the skills and confidence to harness the power of technology to create a better and more inclusive future” (Canada Learning Code, 2021). Since 2011, CLC has offered over 10,500 educational events resulting in over 1.7 million hours spent coding, and engagements with over 600,000 learners across Canada (Canada Learning Code, 2021). The CanCode funding helped CLC develop Learning for the Digital Future: A Pan-Canadian K-12 Computer Science Education Framework (Canada Learning Code, 2020). The framework is meant to provide greater alignment in terms of what Canadian students learn and promote more equitable access to high-quality CS education. The second page of the framework presents the following, as a major rationale for the framework:

> As digital technologies play ever-more important roles in our lives, it is critical that all students, especially those who have been traditionally underrepresented in tech—namely women, visible minorities, Indigenous people, and people living in rural and remote areas—have the opportunity to learn foundational skills and competencies to meet the needs of their time. It is essential that we empower all students to harness the power of these new tools. (Canada Learning Code, 2020, p. 2)

Included in CLC’s framework is a list of 27 other organizations that have been provided with funding from the federal CanCode initiative. In considering the organizations that were provided with this funding, and in considering the general approach used by the federal government to broaden K-12 CS education in Canada, two important issues arise: 1) the provision of funds to non-profit organizations rather than to public education authorities, and 2) the criteria used to assess the success of the initiatives.

To qualify for CanCode funding, organizations must be a not-for-profit organization incorporated in Canada and must have a minimum of three years of experience in the delivery of coding and digital skills programs to K-12 youth and/or teachers (Government of Canada, 2019b). While it was encouraged that the organizations deliver content that maps to provincial/territorial educational curricula, and while it was encouraged that the organizations partner with groups such as public school boards, neither of these criteria were mandatory. These distinctions are important as it signals that not-for-profit organizations, rather than public institutions, were selected to obtain the financial resources to lead CS education initiatives. An alternative approach would have been an investment into the broadening of CS education through groups such as Universities, Colleges, or K-12 Ministries of Education, school boards or schools. One reason why this is so important is that not-for-profit and public educational organizations may embody different approaches, philosophies and end goals related to the broadening of CS education. Not-for-profit organization initiatives might embody an economic argument for broadening CS education, which is workforce centered, 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 (Passey, 2017). Public education organizations, on the other hand, might embody some of the theoretical approaches explored in Chapter 2 of this thesis, such as encouraging Computational Fluency (Resnick, 2017), Computational Participation (Kafai, 2016), or Computational Literacy (diSessa, 2018). Referring back to Table 2, from Chapter 2 of this thesis, an in-depth analysis of the approaches used by not-for-profit organizations associated with the CanCode funding may reveal that their initiatives embody Wing's CS as a topic of study in and of itself, rather than embodying approaches that view CS-related concepts and skills as a tool in mathematics and science, or one that embodies CS concepts and skills for their social, personal, and cultural benefits. Although an in-depth analysis of goals and underlying philosophy of the CanCode initiative is not a part of this chapter or thesis, considering the large amount of federal money associated with the program, it introduces an interesting topic of research centered around who is given the power, control, and resources associated with large scale, CS education initiatives.

As discussed above, it has been reported that in its first two years, the CanCode program had provided more than 800,000 K-12 students and 40,000 teachers with opportunities to learn coding and related skills (Department of Finance Canada, 2019). These numbers included 350,000 girls, over 68,000 Indigenous students, over 100,000 youth at risk, and 34,000 newcomers to Canada (Government of Canada, 2019a). It would be important to investigate the extent of these opportunities and whether or not they involved in depth and prolonged CS education experiences. Likewise, it would be worthwhile to determine the details and extent of the 600,000 “engagements” reported by CLC, or the 1.7 million hours of coding (Canada Learning Code, 2021). The CanCode initiative and the work of the selected organizations can be a valuable means of broadening CS education in the K-12 grades, but a further analysis could help identify the most and least successful components, and whether or not the initiatives led to substantial and effective change. This could help each organization better plan future initiatives, and it could help the government determine criteria for future funding. It would also be interesting to explore the specific grades or concepts and skills that were the focus of the implementation of the CanCode programs, and whether or not these programs reflected curriculum in the various jurisdictions.

4.2.2 United States of America

The broadening of CS education in the United States was exemplified in January 2016 when then President Barack Obama announced a national initiative called *CS For All* that would “empower all American students from kindergarten through high school to learn CS and be equipped with the computational thinking skills they need to be creators in the digital economy” (Smith, 2016, para. 1). The CS For All initiative allocated $4 billion in funding for states and $100 million for school districts that would allow for the expansion of CS teacher training, access to high quality materials, and the development of regional partnerships. The initiative also sought to involve more governors, mayors, and education leaders to help boost CS education and mentions the states of Delaware, Hawaii, Washington, and Arkansas as places where the effective expansion CS opportunities for students had already taken place. The approach of the CS For All initiative, in contrast to Canada’s CanCode project, appears to connect more directly to educational organizations as states and school districts play a more central role. This ensures that educational organizations and experts have more power, control, and potential funding to help shape the direction of the initiatives.

In terms of the foundational goals of the program, the main impetus for the CS For All initiative seems to have been a need to fill current and projected high-tech jobs, and to ensure that all students have access to CS education. In September 2016, the White House released a fact sheet that identified progress related to the CS For All initiative. Thirty-one states were now allowing CS to count towards high school graduation and over 100 organizations had pledged more than $250 million to support CS education. Major support came from the Girl Scouts of the USA, which had the potential to introduce 1.4 million girls per year to CS education, and Code.org which supported the professional development of over 40,000 additional teachers (The White House President Barack Obama, 2016). Also in September, 2016, the K-12 Computer Science Framework (K-12 Computer Science Framework Steering Committee, 2016) was released, which was developed by five different organizations (the Association for Computing Machinery, Code.org, Computer Science Teachers Association, Cyber Innovation Center, and National Math and Science Initiative) in an effort to provide guidance to States and local education agencies as they adopt policies and key infrastructure surrounding CS education. The inclusion of education organizations within the development of this framework, such as the Computer Science Teachers Association and the National Math and Science Initiative, is important to note, as it ensures strong representation from education stakeholders. This recognizes the value that these organizations bring to the broadening of CS education and provides them with more control and influence.

The K-12 Computer Science Framework organized a progression of learning, from kindergarten to grade 12, that centered around five core concepts (computing systems, networks and the internet, data and analysis, algorithms and programming, and impacts of computing) and seven core practices (fostering an inclusive computing culture, creating computational artifacts, collaborating around computing, testing and refining computational artifacts, recognizing and defining computational problems, communicating about computing, and developing and using abstractions).

The intention of the concepts and practices in the K-12 Computer Science Framework was to "serve as a foundation from which all states, districts, and organizations can develop CS education standards for K-12 students" (K-12 Computer Science Framework Steering Committee, 2016, p. 125).

4.2.3 England

England provides an additional, international example of the broadening of CS concepts and skills in K-12 education over the last few years. This approach focusses on incorporating CS concepts and skills within revised curriculum in the public education system, rather than on providing funds to organizations to deliver camps, workshops, webinars or resources. In 2012, the UK's Royal Society published Shut Down or Restart? The Way Forward for Computing in UK Schools which concluded that the delivery of computing education, through existing Information and Communication Technology curricula, was unsatisfactory. Many students were not inspired by what they were taught and were learning little more than basic digital literacy skills such as word-processing and database management. The report identified a specific need to recognize CS as a rigorous academic discipline that is of great importance to the future of all students. Two years later, in 2014, England released a National Computing curriculum that allowed for students, from the age of five, to learn about "the principles of information and computation and how digital systems work" (The Royal Society, 2017, p. 17). The National Computing curriculum includes three major strands: information technology, digital literacy, and CS. The curriculum also identifies CT as a core component of the curriculum and is mentioned in the very first sentence of the National Curriculum document: "A high quality computing education equips pupils to use computational thinking and creativity to understand and change the world" (United Kingdom Department for Education, 2013, p. 1). Examples of subject content for key stage 1 (the first two years of formal schooling, when students are approximately 5 to 7 years old) include understanding what algorithms are and creating and debugging simple programs. Examples of subject content for key stage 2 (years 3 to 6, when students are approximately 7 to 11 years old) include creating programs with specific goals in mind, using sequence, selection, repetition and variables in programs, and using logical reasoning to explain how algorithms work.

A few years after the release of the National Computing Curriculum, the UK's Royal Society published After the Reboot: Computing Education in UK Schools (2017) that provided a list of changes that had taken place since 2012, examined the impact of these changes, and identified "urgent challenges that governments, industry and school leaders need to address in order to safeguard our future efficacy in the digital world" (p. 7). One of these five major challenges was improving the gender balance in computing.

4.2.4 An Important Note from the Author

Considering my experience as a CS teacher and the contemporary literature in the field, it is clear that issues related to equity, diversity, and inclusivity are important to include in a thesis that explores the broadening of CS education in the K-12 grades. While I do not fit the description of a traditionally underrepresented individual in CS, I am committed to furthering my understanding of the issues surrounding the underrepresentation of individuals and groups in CS. With 18 years of experience in K-12 CS education, I observed this lack of diversity in each class that I taught or supported, and was driven to explore interventions, from the small to the large scale, from classroom activities to provincial policy. As I continue my research in K-12 CS education, I have been fortunate to listen to, and learn from, a number of passionate and talented researchers included in this chapter, and I would suggest readers explore their work. I have also been hesitant to take up space in this area, and I am grateful for the conversations I have with my wife Lisa, who does extensive work in K-12 education and research, and with other members of the University community. Since completing the research and analysis for this chapter, I now have a daughter, and this has provided me with an additional perspective. Although I recognize that I have much to learn, these conversations, experiences and perspectives have helped me to better understand my potential allyship role and to continue to actively work towards and support an equitable, diverse and inclusive landscape in CS education.

I am aware that the issue of underrepresented groups in CS goes beyond gender, that complex historical, structural and systemic forces are at play, and that an understanding of intersectionality can provide valuable insight into the analysis of equity, diversity, and inclusivity concerns. There are a number of researchers and authors who have published, and continue to publish, important work in these areas, and I am grateful to have read some of their contributions and will continue to seek out additional voices, research and learn from their lived experiences. In addition, the terminology used when investigating these issues is important. This chapter makes reference to female and male students several times. The rationale for terminology is for clarity and consistency, as these terms are used in the enrolment data obtained from Ontario's Ministry of Education.

Finally, before further exploring equity, diversity, and inclusivity issues and enrolment in CS courses, it is important to identify and discourage a deficit perspective (Patitsas et al., 2014; Vakil, 2018). When discussing underrepresented groups in specific subject areas and disciplines, there is a danger of focusing on a need to "fix" what may be thought of as deficiencies in attitudes, skills, practices, interests, or aspirations. This approach fails to appropriately consider the social structures and systemic issues that may cause inequalities in the first place.

4.3 Potential Impact and Missed Opportunities

A central and consistent theme to many of the programs meant to broaden CS K-12 education has been the concern surrounding underrepresented groups in CS education and the field. Canada's 2021 federal budget indicates that the CanCode initiative has a "special focus on reaching young people who are traditionally underrepresented in science, technology, engineering and mathematics, such as girls and Indigenous youth" (Department of Finance Canada, 2021, p. 115) while the K-12 Computer Science Framework recognizes the opportunity gap that exists when there is a disparity in access to CS education, as often traditionally underrepresented students, who already face educational inequities, are further marginalized (K-12 Computer Science Framework Steering Committee, 2016). While the research proves that there are a number of diverse and intersecting groups traditionally underrepresented in CS Education, this chapter focuses specifically on the concern surrounding the underrepresentation of female students.

There are three main concerns surrounding the underrepresentation of female students in secondary school CS courses. The first involves the missed economic opportunities that would be afforded to students if they chose CS as a field of study and career. Excluding female students from CS education means excluding them from lucrative, high-status and flexible careers within the continually growing field of technology (Information and Communications Technology Council, 2017). In addition, CS concepts and the thought processes involved in computational thinking are recognized as valuable for all students to learn (Wing, 2006) as they are applicable to a wide range of careers. Scientists and researchers in biology, chemistry, physics, astronomy and medicine use computers and CS concepts for mathematical modelling in order to expand the frontiers of knowledge in both research and innovation (Monroe, 2014) while CS-related, transformational technology such as virtual and augmented reality, 5G mobile, 3D printing, blockchain and artificial intelligence will lead to an increased demand for digital skill workers in lucrative fields (Information and Communications Technology Council, 2017). The second concern surrounding the underrepresentation of female students in CS involves CS knowledge serving as a critical part of being an educated, 21st century citizen (Margolis et al., 2012). Resnick (2017) explains:

> In today's society, digital technologies are a symbol of possibility and progress. When children learn to use digital technologies to express themselves and share their ideas through coding, they begin to see themselves in new ways. They begin to see the possibility for contributing actively to society. They begin to see themselves as part of the future. (p. 50-51)

Kafai's (2016) concept of Computational Participation acknowledges that the thought processes associated with CS are a social practice with a broad reach. Rather than an abstract discipline, programming is now a way to make and be in the digital world (Kafai, 2016). Digital technologies are used for functional, political, and personal reasons and therefore all students should develop an understanding of interfaces, technologies, and systems that they encounter on a daily basis.

Finally, an underrepresentation of female students in CS leads to the field missing out on the benefits that a diverse labour force can have on innovation. Computer scientists help design tools that shape modern society and diversifying the field means a higher likelihood of creating technologies that are appropriate for a broad population (Margolis & Fisher, 2002). CS based technologies can also help solve economic, environmental, political, and social problems and thus diverse perspectives in the field are needed in order to develop diverse solutions. A lack of diversity in high school CS courses could lead to a lack of diversity in the technology sector and therefore a lack of diversity in solutions and technologies available for all.

Considering the above-mentioned impact and missed opportunities resulting from a lack of diversity in CS education, it is important to develop an understanding of enrolment patterns in Ontario Computer Studies courses while the numerous initiatives related to the broadening of CS education have been taken place. It is also important to investigate, specifically, whether or not a gender gap currently exists in Ontario's high school CS courses. Before doing so; however, a review of equity, diversity, and inclusivity issues in CS is provided, as well as important and relevant conceptual frameworks related to gender equity and CS education.

4.4 Equity, Diversity, and Inclusivity in CS Education

Before investigating enrolment data and analyzing Ontario's secondary Computer Studies courses, it is important to develop an understanding of the wide range of issues related to gender, equity, diversity, and inclusivity in CS education that span historical, political, psychological, and social domains. Acknowledging the complexity of these issues, Patitsas et al. (2014) employed a historical sociology approach that argued that for educators to understand the current situation and how to change it, they must understand historical forces: "we cannot reduce the matter down to a few issues that, if fixed, would change everything" (p. 111). As well as appreciating the complexity of the issues, several authors also caution against the tendency to employ a deficit approach when discussing the CS education gender gap.

Victores and Gil-Juárez (2017) explain that too often girls are portrayed as deficient and lacking the “normal relationship” (p. 671) that boys and men have with computing. Focusing on a need to “fix” the deficiencies of girls’ attitudes, skills, practices, interests, and aspirations in computing fails to appropriately consider the social structures that may cause these inequalities in the first place. Consideration of this deficit approach is important, as interventions implemented to increase enrolment of female students in CS courses should certainly avoid an approach that considers needing to fix attitudes or aspirations. Gender differences in the attitude towards technology use has often been cited as an important factor in explaining the underrepresentation of female students in CS education, however; findings surrounding this issue have been inconsistent (Cai et al., 2016).

Cheryan et al. (2015) emphasize the impact that stereotypes can have on signaling to girls that CS is not an appropriate field for them. The authors explain that students have stereotypes about the culture of CS and that girls face negative stereotypes about their abilities in the field. These stereotypes include girls being steered away from CS by parents and teachers who consider the field more appropriate for boys (Eccles et al., 1990; Sadker & Sadker, 1994), as well the underrepresentation of female students in CS perpetuating future underrepresentation (Murphy et al., 2007). In addition, the authors note that girls underestimate their potential level of achievement in the field (Correll, 2001; Ehrlinger and Dunning, 2003) and they may anticipate greater work-family conflicts in CS than they would in other fields (Ceci et al., 2009). The authors also acknowledge the fact that there is gender discrimination in the CS field, reduced opportunities for women, and social and professional penalties for women when exhibiting competence and leadership qualities in CS-related occupations (Moss-Racusin et al., 2012; Rudman, 1998). If interventions meant to reduce gender gaps in CS education are to be effective, the general state of the field itself, and the work environments associated with CS need to be considered. It is not enough to increase young women’s participation in CS at the secondary level, only for these students to eventually find gender discrimination in the field.

When exploring the psychological explanations for why girls avoid computer-related subjects, Vitores and Gil-Juárez (2016) explain that girls have negative stereotypes of computer scientists being geeky and the field being male dominated and oriented towards working with machines rather than people. They also acknowledge that girls have poor knowledge of CS as a discipline and career and that they often perceive CS as being boring. While boys also share this negative view, Fisher and Margolis (2003) and Lang (2010) indicate that this belief is more damaging for girls than boys. It is here that work surrounding appropriate curriculum and pedagogy can potentially have an impact. There are a number of approaches that can be taken towards K-12 CS education, many of which were summarized in Chapter 2. A curriculum and related pedagogy that focuses on CS as a topic of study in and of itself, such as one presented by Wing (2006), may be less effective at engaging students from underrepresented groups. Instead, an approach that recognizes the power of the computer as a tool, such as Papert (1993) or diSessa (2018), or that incorporates the social, personal and cultural contexts, such as Kafai (2016) and Resnick (2017) may be more effective.

Adding to the literature surrounding large scale initiatives meant to broaden CS education participation, Vakil (2018) calls for a justice-centered approach to equity in CS:

> With CS rapidly emerging as a distinct feature of K–12 public education in the United States, calls to expand CS education are often linked to equity and diversity concerns around expanding access to girls and historically underrepresented students of color. Yet, unlike other critical traditions in education research, equity-oriented CS research has largely failed to interrogate the sociopolitical context of CS education. (p. 26)

Vakil's (2018) justice-centered approach attempts to move away from what he calls the dominant approach for broadening CS concepts and skills, which may be in the best interests of multinational corporations, towards focusing on "the sociopolitical implications, relevance, and, ultimately, liberatory possibilities of teaching and learning CS" (p. 27). In the dominant approach, students are encouraged to be responsible digital citizens, to potentially pursue career opportunities, and the role of student identity in learning processes is undertheorized, “resulting in deficit lens on girls and students of color” (p. 37). Alternatively, a justice-centered approach includes students moving beyond being responsible digital citizens, to students engaging in critiquing unethical abuses of technological power, while researchers consider learning environments that are responsive to students’ multiple social identities. In a justice-centered approach, CS learning is framed as being “important for the social and economic welfare of historically nondominant students and their communities”, as students are encouraged to “pursue CS as part of and connected to larger struggles for justice and liberation” (p. 37). Vakil’s justice-centered approach uses critical pedagogy and critical race theory as conceptual frameworks to situate his work. The conceptual frameworks for situating this chapter’s analysis of enrolment follow.

4.5 Conceptual Frameworks

Gender has always been a central theme in the organization of education. Discourses have often been informed by the meanings that have been given to identifying students as either male or female, as well as the biological and hormonal differences between these two categories (Pinar et al., 1995a). In the 1970s, the understanding of curriculum as a gender text became an important form of analysis that was founded on feminist theories developed during curriculum’s Reconceptualization period. The analysis below provides an introductory look into feminist theory and its value in informing an investigation into gender gaps and the underrepresentation of female students in high school CS.

In the late 1960s Joseph Schwab, a leading figure in the curriculum based educational reform movement (Connelly, 2013), declared the state of curriculum studies ‘moribund’ and called for new principles and new methods of analysis (Deng, 2018). What followed was a transition from an interest in the development of curriculum to a theoretical and practical interest in understanding curriculum (Pinar et al., 1995b). Tyler’s Rationale, which outlined a practical four step process of curriculum development that included stating objectives, selecting experiences, organizing experiences, and evaluating (Kliebard, 1970), had dominated as an approach to curriculum studies but it had now reached the end of its utility. The Reconceptualization period of curriculum studies involved moving away from what was considered an atheoretical, practical approach to one of understanding that borrowed modes of inquiry that were historical, philosophical, and literary and that were popular in humanity fields (Pinar, 1975). One of these modes of inquiry that developed to inform curriculum studies was that of feminist theory.

Feminist theory focuses on analyzing gender inequality and socio-political structures. It recognizes that what were once thought to be “humanly inclusive problematics, concepts, theories, objective methodologies, and transcendental truths” (Harding, 1986, p. 15) are instead, products of thought whose creators were marked by gender, class, race, and culture. Sometimes referred to as an emancipatory epistemology, feminist theory challenges conceptual frameworks in a wide variety of fields and seeks to ask questions related to the influences and impact of androcentric points of view.

A feminist approach to research related to the underrepresentation of female students in high school CS is appropriate, however; it is important to first develop an understanding of some theoretical implications. Initially, the liberal feminism of the 70s and 80s was focused on getting more women to enter the science and technology field and as a result suggested that the gender gap could be fixed through socialization and equal opportunity policies (Wajcman, 2007). It was believed, however; that his approach situated the problem within women as they were being asked to change major aspects of their gender identity and forsake their femininity. Later, socialist and radical feminists explored further the gendered nature of technoscientific knowledge and culture and the gender power relations that were embedded in the science and technology fields (Wajcman, 2007). Unfortunately, some of these approaches presented a negative image of women as victims of a patriarchal technoscience and neglected to recognize the agency that women had and the potential of redesigning technologies for gender equality. Presently, researchers such as Vitores and Gil-Juárez (2016) warn against falling into these theoretical traps. They caution against the paradox of reproducing dangerous assumptions about computing and gender through research that hopes to identify and solve the problem in the field. As a solution, they highlight the need for “different researchers’ eyes” that would allow for varied landscapes in the field of women in computing research including acknowledging the limitations of gender binaries (Henwood, 2000) and the black-boxing of gender (Grint & Gill, 1995). Two such sets of eyes include technofeminism and material feminism.

Technofeminism borrows from feminist and technology studies and seeks to disrupt the idea that technology is a product of “rational technical imperatives” (Wajcman, 2007) and instead argues that it is a source and consequence of gender relations that are in constant flux. A technofeminist approach recognizes that gender is understood as a “performance or social achievement, constructed in interaction” (Wajcman, 2007, p. 294) and that relationships between gender and technology are not fixed across time and location. As an example, a smart phone may serve as a liberating extension of a Western women’s body or as a tool that allows for her mother to keep track of her daughter. In Bangladesh, however; the smart phone serves as a communication device allowing women traders to run their business, and in Central Africa the smart phone is a source of military conflict involving scarce minerals that affect women in the surrounding area. “There is enormous variability in gendering by place, nationality, class, race, ethnicity, sexuality and generation and thus women’s experience of ICTs (information communication technologies) will be diverse” (Wajcman, 2007, p. 294). Technofeminism’s central premise is that people and objects co-evolve, resulting in a need for new perspectives on research surrounding women in computing education that acknowledges the sociotechnical networks and systems at play. Similarly, material feminism provides a valuable theoretical framework that sheds light on the complex social dynamics involved in gender and technology research.

Material feminism arose as a result of concerns surrounding postmodern feminism’s epistemology suggesting that the real and material is a product of language (Alaimo & Hekman, 2008). While the associated linguistic and discursive turn of postmodernism was productive, defining such things as materiality, the body and nature as products of discourse failed to take the more-than-human world seriously (Alaimo & Hekman, 2008), a fact that had important implications for the study of gender and science. Material feminism, by contrast, acknowledges nature as more than a passive social construction and instead, a force with agency that interacts with and potentially changes other elements in a network, including humans. Material feminism resists returning to modernism by avoiding the dichotomy between construction and reality and instead acknowledges that while language does construct reality, it also interacts with other elements in this construction (Hekman, 2008). Moving from an epistemological perspective to an ontological one, Hekman (2008) explains that feminism requires an understanding that concepts and theories have material consequences: “There is a world out there that shapes and constrains the consequences of the concepts we employ to understand it” (p. 109).

Considering the materials, objects and technology inherent in the field and education of CS, both technofeminism and material feminism provide important conceptual frameworks with which to approach the theme of equity, diversity, and inclusivity. This theme will now be explored through an analysis of both general enrolment data from secondary CS courses in Ontario, and through the enrolment data related specifically to female and male students.

4.6 Enrolment in Ontario Secondary School Computer Studies

In order to obtain a Secondary School Diploma in the province of Ontario, students must earn 18 compulsory and 12 optional credits as well as pass the provincial literacy requirement and perform a minimum of 40 hours of community involvement activities (Ontario Ministry of Education, 2015). Of the 12 optional courses, at least one must come from a group of subjects that include science (grade 11 or 12), technological education, French as a second language, computer studies or cooperative education.

The current computer studies curriculum includes a total of five courses distributed over grades 10, 11 and 12 (Ontario Ministry of Education, 2008). These courses include:

- ICS2O: Grade 10 Introduction to Computer Studies – Open;
- ICS3C: Grade 11 Introduction to Computer Programming – College;
- ICS3U: Grade 11 Introduction to Computer Science – University;
- ICS4C: Grade 12 Computer Programming – College; and
- ICS4U: Grade 12 Computer Science – University.

The courses are classified as either open, college preparation or university preparation. The open courses are designed to broaden students’ knowledge and skills in computer studies while the college preparation courses are designed to equip students with the knowledge and skills to meet program requirements for college, apprenticeships, or other training programs. The university preparation courses are designed to equip students with the knowledge and skills required to meet university program requirements. Of the five courses, only two require prerequisites: students must have obtained the ICS3C credit in order to enroll in ICS4C and they must obtain the ICS3U credit in order to enroll in ICS 4U. The grade 10 ICS2O course is not a prerequisite for either the grade 11 ICS3C or ICS3U course.

4.6.1 Overall Enrolment

Total student enrolment data for Ontario’s five secondary Computer Studies courses was obtained online through Ontario’s Data Catalogue. The Ontario Data Catalogue includes thousands of data sets including enrolment data for all of Ontario’s secondary school courses. The course enrolment in secondary schools data includes the number of students enrolled in ministry defined secondary school courses and includes the course code, course description, grade, pathway or destination (such as College or University) and the number of students enrolled for each course.

The data for the total number of students enrolled in Ontario’s five secondary Computer Studies courses is shown in Figure 2.

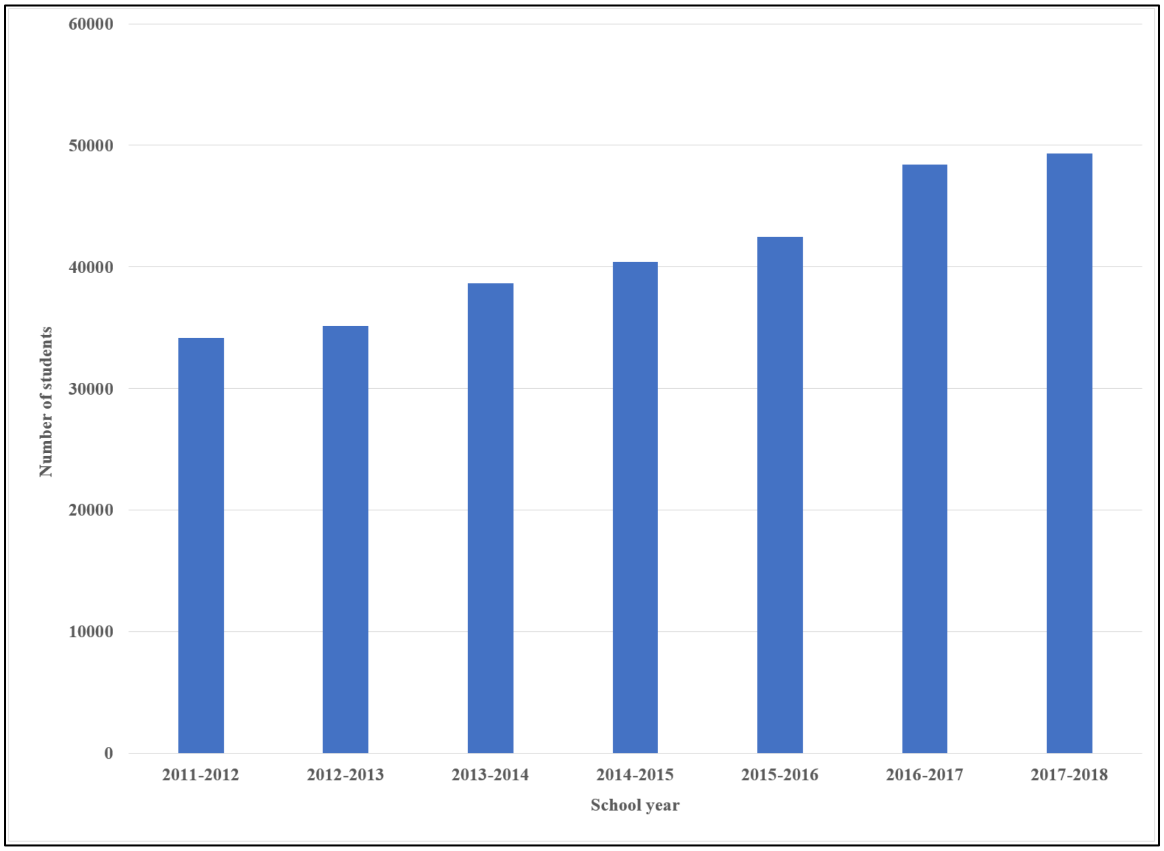


Figure 2. Total number of students enrolled in Ontario secondary Computer Studies courses (2011-2018)

The data reveals that enrolment has increased in Ontario's secondary Computer Studies courses each year, since the 2011-2012 school year. During the 2011-2012 school year 34,177 students were enrolled in secondary Computer Studies courses while in 2017-2018, this number had increased to 49,358.

In addition to the number of students enrolled in Computer Studies courses, the total number of students enrolled in Ontario secondary schools was also obtained from the Ontario Data Catalogue. This data, in combination with the data related to the number of students enrolled in Computer Studies courses, provides the percentage of secondary students in Ontario who are enrolled in Computer Studies courses. This data is shown in Figure 3.

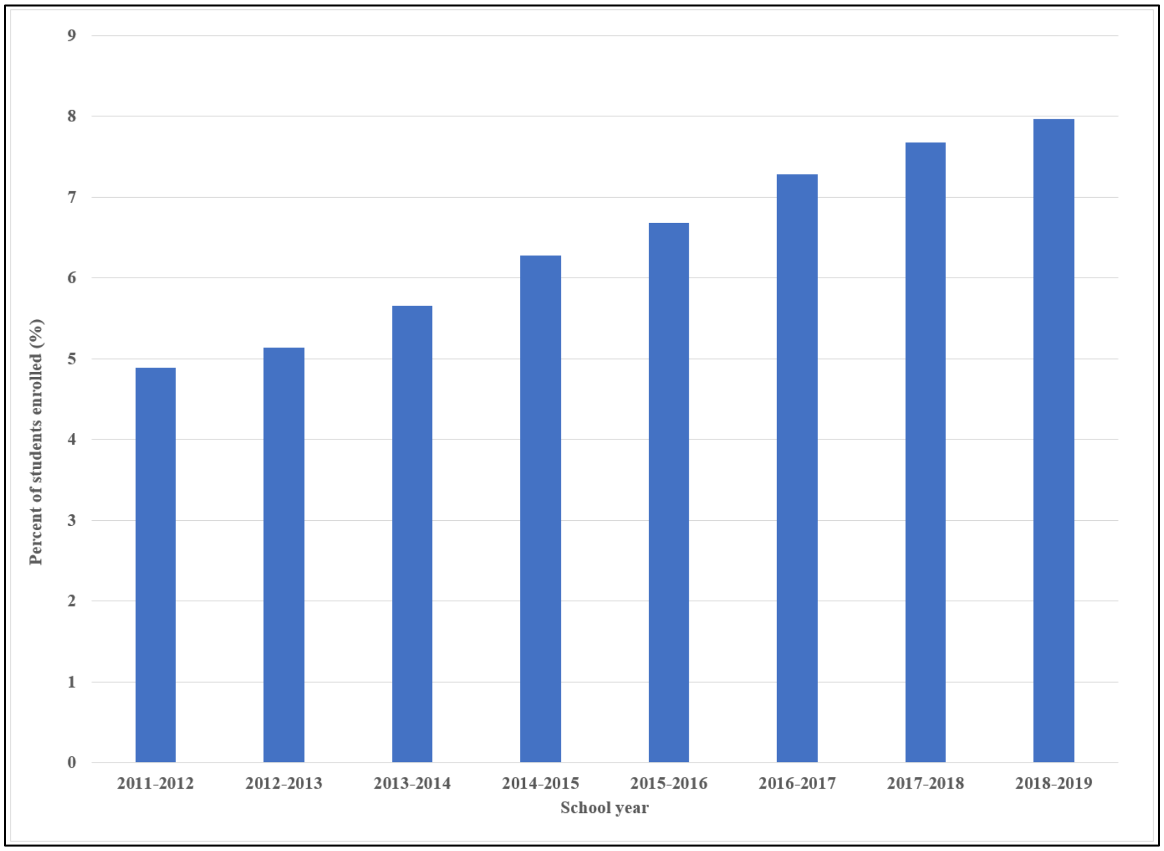


Figure 3. Percentage of secondary students enrolled in Ontario secondary Computer Studies courses

As shown in Figure 3, the percentage of students enrolled in secondary Ontario Computer Studies courses increased each year, since the 2011-2012 school year. During the 2011-2012 school year less than 5% of Ontario secondary school students were enrolled in secondary Computer Studies courses while during the 2018-2019 school year, this number had increased to almost 8%.

A more detailed breakdown of the data provides insight into the enrolment of students within each of the specific Computer Studies courses. Figure 4 indicates the number of students enrolled in the grade 10 ICS 2O course, the grade 11 ICS 3C and 3U courses, and the grade 12 ICS 4C and 4U courses. Figure 5 breaks down the data even further, showing enrollment data for each of the five individual Computer Studies courses.

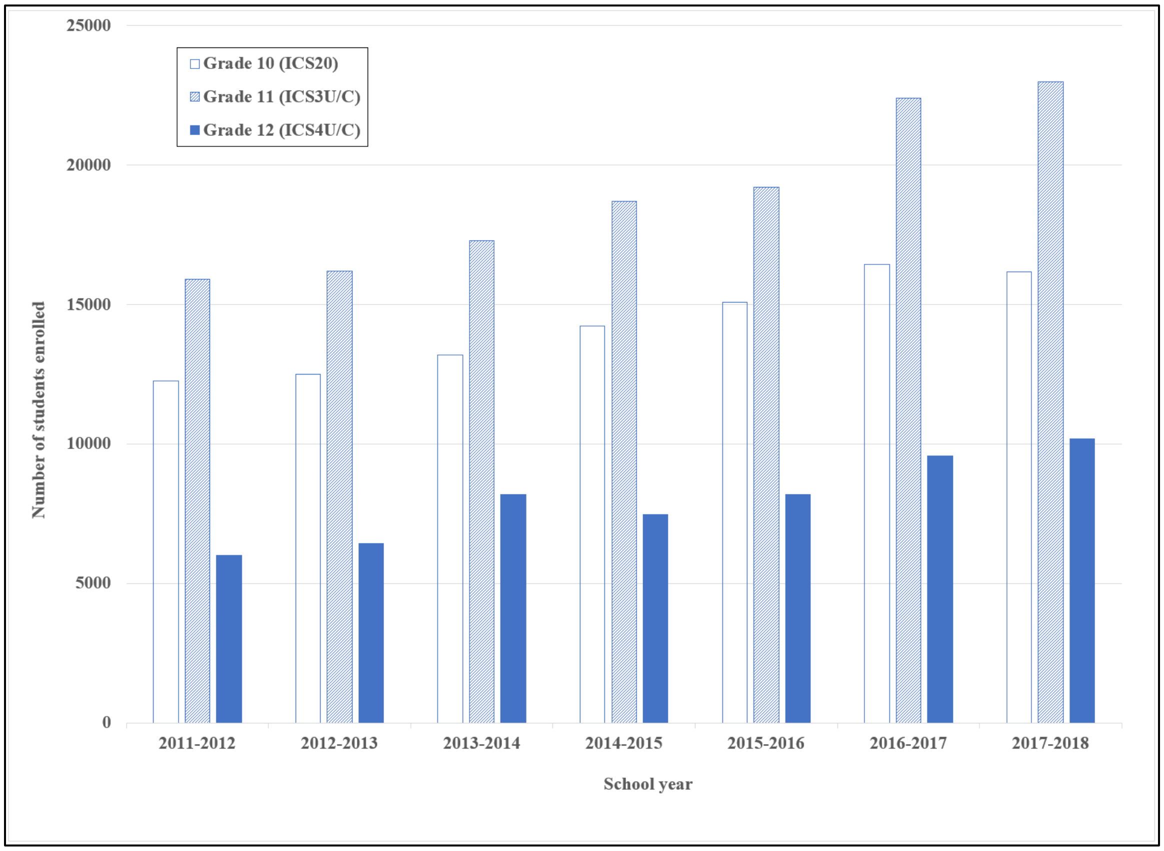


Figure 4. Total number of students enrolled in Computer Studies courses in each grade

It is clear that the grade with the largest number of students enrolled in Computer Studies courses is grade 11, and then there is a significant drop off as much fewer students enroll in the grade 12 courses. It is also clear that the majority of students enrolling in grade 11 or grade 12 courses are enrolled in the University pathway course, and not the College pathway course. In addition, since 2011, there has been very little increase in the number of students enrolling in the grade 11 or grade 12 College pathway courses. The increases in enrollment, from year to year, for the grade 11 and grade 12 courses are related to the increase in the University pathway courses.

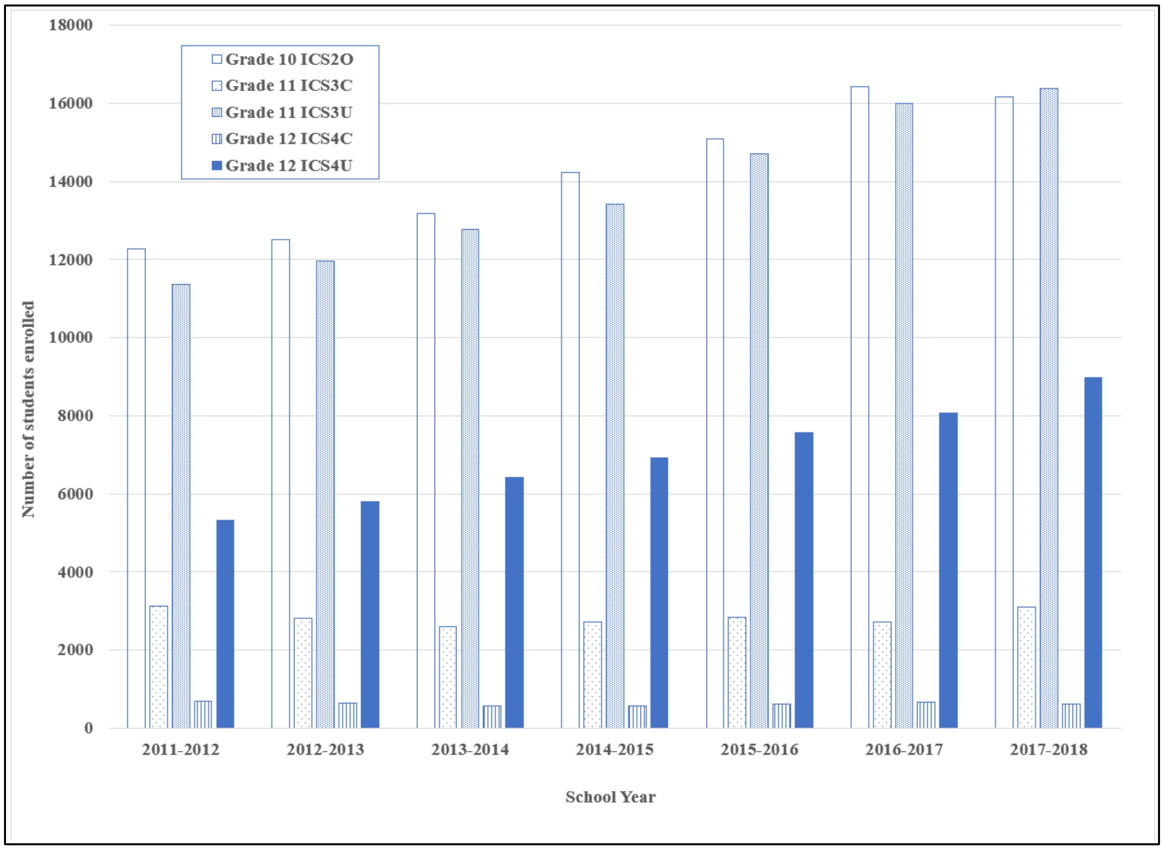


Figure 5. Total number of students enrolled in the five Computer Studies courses

The following provides a summary of findings above, before moving on to specific data related to female and male student enrolment in Computer Studies courses:

- enrolment has increased in Ontario's secondary Computer Studies courses each year, since the 2011-2012 school year;
- the percentage of students enrolled in secondary Ontario Computer Studies courses increased each year, since the 2011-2012 school year;
- the grade with the largest number of students enrolled in Computer Studies courses is grade 11;
- there is a significant decrease in enrolment after grade 11, as fewer students enroll in the grade 12 courses;
- the majority of student enrolling in grade 11 or grade 12 courses are enrolled in the University pathway course and not the College pathway course;

- there has been very little increase in the number of students enrolling in the grade 11 or grade 12 College pathway courses
- the increases in enrollment, from year to year, for the grade 11 and grade 12 courses is largely due to the increase in enrolment in the University pathway courses.

4.6.2 Diversity and Ontario Computer Studies

The data used above was obtained from the Ontario Data Catalogue, which does not provide information related to the enrolment of female and male students in the various secondary courses. This specific data; however, was available under Ontario's Freedom of Information and Protection of Privacy Act and by making a formal request to Ontario's Ministry of Education. The data obtained includes male and female student enrolment in all five courses at public and Roman Catholic schools since the 2009-2010 school year. The data does not include enrolment from private schools and publicly funded hospital and provincial schools, care, treatment, and correctional facilities. It also does not include enrolment from summer, night, and adult continuing education day schools. Figure 6 shows the total number of female and male students enrolled in Computer Studies courses in Ontario.

The data in Figure 6 reveals that there is a disproportionately low number of female students enrolled in secondary Computer Studies courses in Ontario, indicating that a significant gender gap exists. During the 2017-2018 school year, female students made up only 21.5% of secondary school Computer Studies students. This means that there is one female student for every four male students in an Ontario secondary school CS classroom. Considering a class of 25 students, there would only be approximately 5 female students on average.

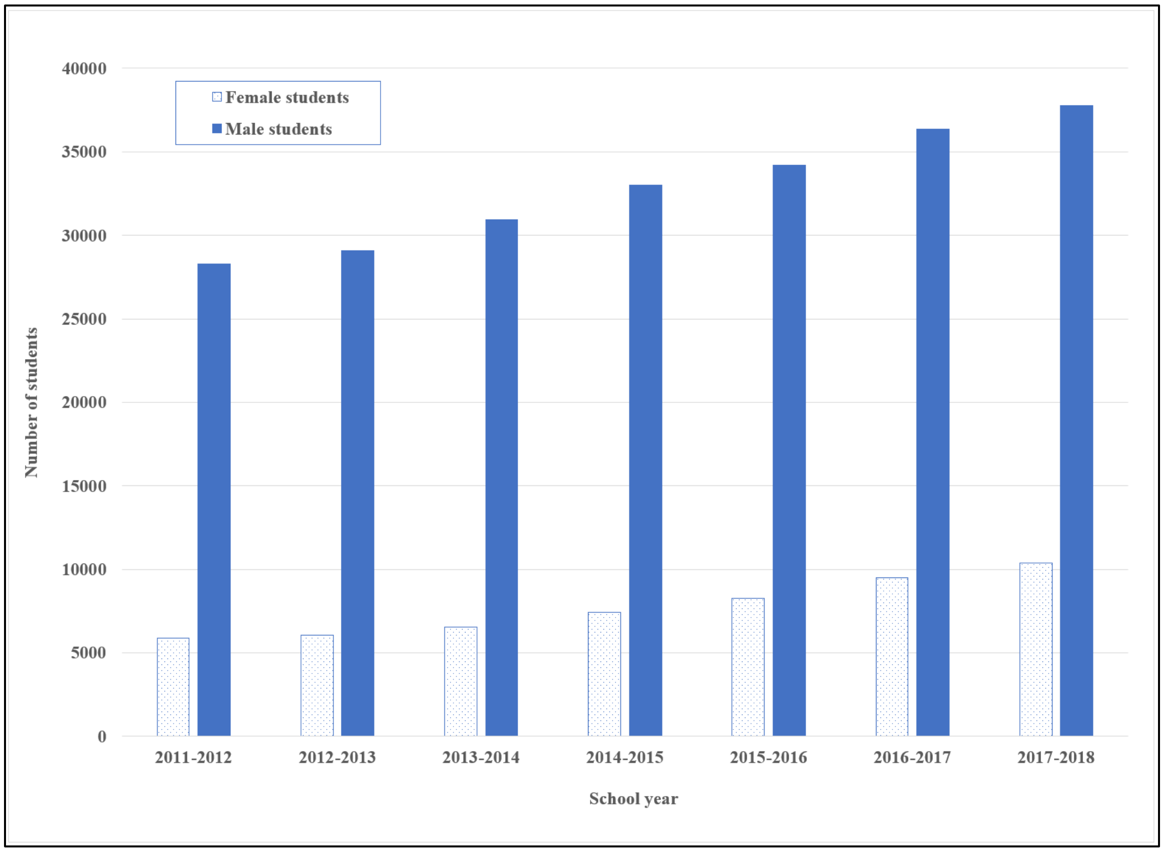


Figure 6. Total number of female and male students enrolled in Computer Studies courses

While Figure 2 indicates that enrolment in CS courses has increased over the last few years, Figure 6 shows that this increase is not equally represented by female and male students. The enrolment of female students in Computer Studies courses, from 2011 to 2018, has increased by 76% while the enrolment of male students in Computer Studies courses, during that same time frame, has increased by 34%.

Figure 7 reveals that female student enrolment in all five of the Computer Studies courses has been increasing since 2010. The most significant increases are in the grade 10 ICS2O and grade 11 ICS3U courses. The grade 12 ICS4U course shows a slight increase; however, both College pathway courses (ICS3C and ICS4C) show little increase in female student enrolment.

Figures 7 shows the percentage of enrolled female students, in each of the five Computer Studies courses, and shows that the percentage of female students in the grade 10 ICS2O, grade 11 ICS3U and grade 12 ICS4U courses have been increasing, while there has been no decrease for the ICS3C and ICS4C courses.

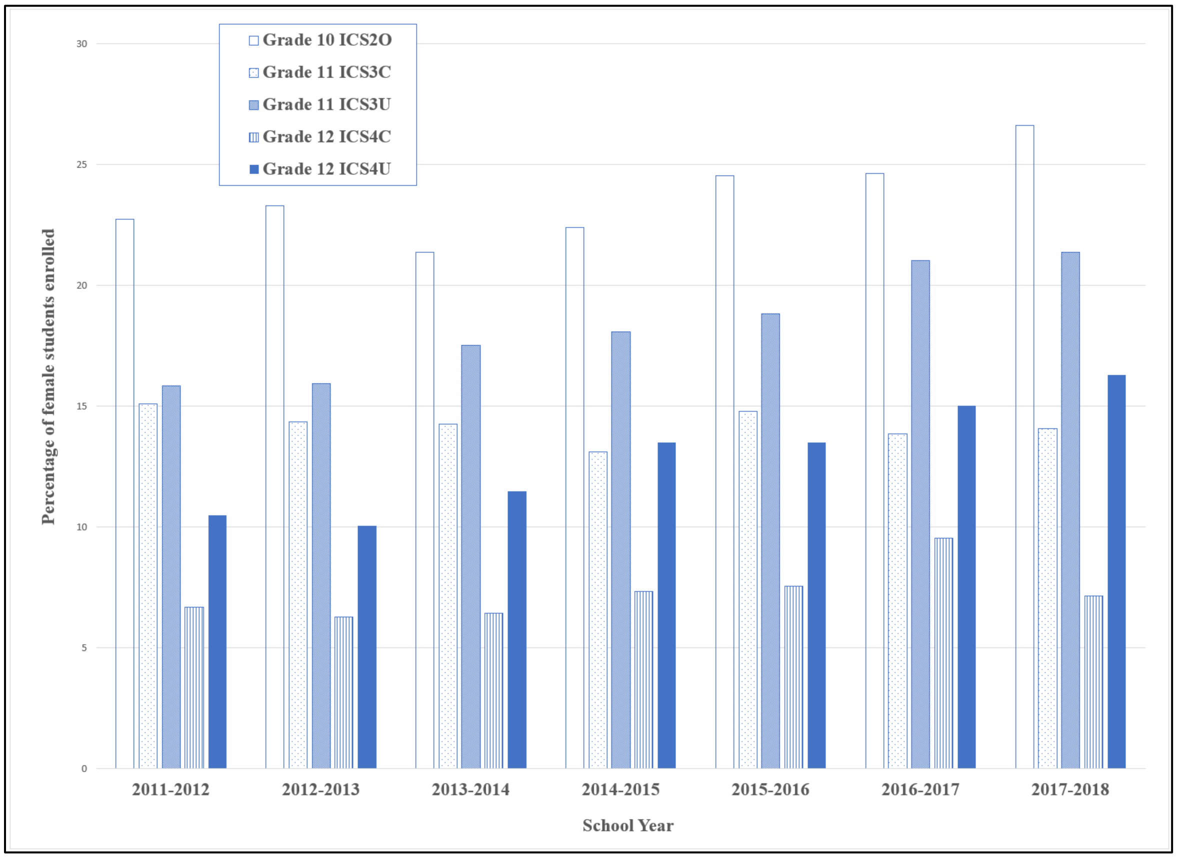


Figure 7. Percentage of female students enrolled in each of the five Computer Studies courses, from 2011 to 2018.

It is also apparent that the percentage of female students who make up the Computer Studies courses decreases in the later grades. In 2017-2018, the percentage of female students enrolled in each of the five courses, is:

- grade 10 ICS2O – 26.6%;
- grade 11 ICS3C – 14.1% and ICS3U – 21.4%;
- grade 12 ICS3C – 7.1% and ICS4U – 16.3%.

Considering the data above, it is clear that in Ontario's secondary, Computer Studies courses:

- there is a significant gender gap with female student enrolment accounting for only 21.5% of secondary school Computer Studies students during the 2017-2018 school year;
- the gender gap has been decreasing since 2011, as female student enrolment has increased by 76% while the enrolment of male students in Computer Studies courses, during that same time frame, has increased by 34%;
- the gender gap is smallest in the grade 10 course, and then increases each year in the subsequent grades;
- the gender gap is greatest in the grade 11 and grade 12 College pathway courses.

As shown in Figure 3, the percentage of students enrolled in secondary Ontario Computer Studies courses has increased each year, since the 2011-2012 school year. During the 2011-2012 school year less than 5% of Ontario secondary school students were enrolled in secondary Computer Studies courses while during the 2018-2019 school year, this number had increased to almost 8%. It is clear that the College pathway courses have the lowest enrolment, indicating a need to further understand the issues related to this phenomenon. It is possible that fewer students choose the College courses when offered, for a variety of potential reasons, but it will also be important to examine whether or not all courses are offered at all schools. It will also be interesting to investigate why there is a decrease in enrolment in the final grade 12 courses. It is possible that students enroll in the grade 10 or 11 courses and then decide not to continue to pursue CS as a possible career direction, but it is also possible that other factors play a role, such as the required courses needed for high school graduation or the courses required for acceptance into University programs.

In terms of data related to female and male student enrolment in Computer Studies courses, it is clear that there is a significant gender gap in secondary CS education in Ontario. Interestingly, the gender gap has been decreasing since 2011. Some of the influences that contribute to a gender gap in CS education were included in earlier sections of this chapter, in order to better understand the context of the study. What follows is a discussion on some of the literature related to large scale frameworks that seek to better understand interventions that have been used to address equity, diversity, and inclusivity concerns in CS education and identify important leverage points that could be used.

4.6.3 The Universal/Selective/Indicative Model and Systems Thinking Leverage Points

The Universal/Selective/Indicated (USI) model, presented by Patitsas, Craig and Easterbrook (2015), is a conceptual tool borrowed from public health's suicide prevention literature (Wasserman et al., 2009) that categorizes initiatives based on their targeted audiences. Universal initiatives are those that are carried out without considering the target groups of the population. Developing a student mentorship program within a CS department, increasing paired (partnered) computer programming initiatives, mandating that all students enroll in a CS course, admission changes or switching to blind review for conference selection are all examples of universal diversity initiatives used in post-secondary CS education that impact entire populations of students but that disproportionally benefit underrepresented groups including women and minorities (Patitsas et al., 2015). In terms of the Ontario context, a universal initiative might be a revision of secondary Computer Studies, in an effort to incorporate broader CS content, skills and connections that go beyond simply the study of the computer in and of itself. This could include curriculum expectations that support cross-curricular projects, creativity, and solving problems within local contexts and communities. Alternatively, adding coding concepts and skills to the K-8 curriculum could also be seen as a universal initiative, as it would impact all Ontario K-8 students, but would also ensure that coding and CS-related concepts are introduced to underrepresented groups at an earlier age.

Selected initiatives include those that specifically target underrepresented groups within a population. Examples of effective selected initiatives used in post-secondary CS include a mentorship program for all female students, departmental women-in-CS clubs, outreach initiatives for girls and scholarships for women in CS (Patitsas et al., 2015). Canada's federal CanCode project supports selected initiatives through its provision of funding to organizations who support the broadening of CS education to specific, underrepresented groups. Examples of these organizations include Black Boys Code, Hackergal and Ulnooweg (Government of Canada, 2020). Black Boys Code focuses on introducing digital literacy and programming skills to black boys ages 8 to 17 years old, while Hackergal focuses on supporting girls between the ages of 11 and 14. Ulnooweg is an Indigenous led initiative that supports Indigenous, First Nation & Metis students from kindergarten to grade 12 (Government of Canada, 2020).

Finally, indicated initiatives are those that target specific individuals who are part of a target group and who may require extra supports. Examples of indicated initiatives that have been effective in post-secondary CS education include a mentorship program developed specifically for students who have been flagged as requiring assistance or when a teacher or adviser recognizes a student who is struggling and provides support or encouragement (Patitsas et al., 2015).

In addition to considering the target groups of diversity initiatives, Patitsas et al. (2015) also encourage educators to consider the leverage of these initiatives by asking: Does the initiative lead to superficial or whole-sale system changes? The authors borrow and simplify Donella Meadows's (2008) leverage-point continuum and identify four categories of leverage based on the Structure-Behaviour-Function Theory (Hmelo-Silver & Pfeffer, 2004). These categories include, from smallest to greatest leverage: structural changes, system behaviour change, function change, and paradigm change. Structural changes that have proven to be effective in improving the number of female students in post-secondary CS education include having the introductory course taught by a female instructor, using female pronouns in assignment instructions, assigning groups based on gender and providing multiple entry points into a CS major. System behaviour changes that have proven to be effective in improving the number of female students in post-secondary CS education include improved research opportunities, using meaningful contexts for assignments, using blind reviews for scholarship applications, and removing potential stereotypes (such as androcentric posters). Function changes that have proven to be effective in improving the number of female students in post-secondary CS education include outreach efforts, increased feedback to students, altering program entry requirements, and new classroom rules such as calling on students randomly. Finally, paradigm changes that lead to the greatest leverage in improving the number of female students in post-secondary CS education include shifts in thinking that identify the problem within the system rather than the individual, teaching in a way that empowers the students, and viewing computing excellence as something that can be taught rather than something that is connected to an individual's innate ability.

The USI and leverage frameworks presented by Patitsas et al. (2015) serve as important tools that allow researchers to zoom in on diversity initiatives and evaluate their effectiveness within large educational systems. In contrast, educators, policy makers and researchers also need to be able to zoom out, away from specific initiatives and implementation models, towards relevant theoretical frameworks that are useful for situating and informing this important and complex work within appropriate epistemological and ontological grounds. The technofeminist and material feminist frameworks presented earlier provide these grounds.

4.6.4 Margolis and the Clubhouse Today

A major impetus for this chapter began with Margolis and Fisher's (2002) work, so it seems appropriate to conclude with some updated and contemporary views of the issue from Margolis herself, as well some of her co-authors and colleagues. In 2015, Margolis, Goode and Chapman (2015) acknowledged that a number of education stakeholders in the US, including nonprofits, industry partners, politicians, school districts and parents, were beginning to raise concerns about the importance of access to K-12 CS education. Based on their work with the Exploring Computer Science (ECS) program, the authors warned, however; about the superficiality of numbers: "focusing on quantitative metrics sometimes provides little more than a head-count of students enrolled in course. It does not tell us if students are prepared, engaged, and challenged with computing, or disengaged and marginalized" (p. 60). This is an important consideration for large initiatives such as Canada's CanCode and the US's CS For All programs. Simply recording "engagements" and "coding hours" does not provide any information related to the depth and type of CS education experienced by students involved in the programs. Nor does it capture data on the longer lasting impact of these initiatives, and whether or not they lead to more students from underrepresented groups enrolling in CS courses, and potentially following pathways leading to a career in the field.

A potential solution is a set of instruments developed through the ECS program that assess student's engagement, attitudes, interest, and ability to apply, evaluate and explain what they are learning. In addition to acknowledging the importance of a program evaluation tool that goes beyond simple enrollment numbers, the authors also provide some lessons from scaling. These lessons explore the "tight but loose" tension, described by Thompson and Wiliam (2017), that must be navigated when large scale initiatives must remain faithful to an original model and true to original values (tight), while also providing flexibility to meet the needs of local conditions (but loose). Some lessons from scaling include the need for:

- teachers to promote cognitively challenging discussions;
- continuous professional development and professional community building for teachers;
- continuous technical assistance and support for teachers; and
- ongoing relationship building, communication and advocacy amongst stakeholders.

In addition to these lessons, the authors also warn against potential unintended consequences of scaling initiatives, and they provide the example of how the expansion of the ECS program lead to closer scrutiny of teacher certification regulations and a stalling of initiative momentum.

Finally, considering the important contributions that Margolis and Fisher's (2002) seminal book made to the awareness of issues concerning female student enrollment in CS courses, it is also important to expand the scope of investigation and consider other underrepresented groups in CS education. In Stuck in the Shallow End: Education, Race, and Computing, Margolis et al. (2008) investigate a lack of access to high school CS courses based on influences associated with race and socioeconomic status. Much like Unlocking the Clubhouse, Stuck in the Shallow End begins with an investigation surrounding specific groups and CS access, but serves a much larger purpose as a "treatise on the potential and reality of education to remove barriers and to support social and economic equality" (p. vii).

4.7 Conclusion

An analysis of enrolment data from secondary Computer Studies courses in Ontario indicates that overall student enrolment has increased since the 2011-2012 school year. This increase has been primarily due to an increase in enrolment in the grade 10 ICS2O course and the grade 11 and grade 12 University pathway courses. It is evident that further research is required related to the College pathway courses, as these courses have significantly lower enrolment than the others and there has been very little increase in enrolment in these two courses over the seven years studied in this work. The data also reveals a gender gap in Ontario secondary Computer Studies courses, as female students make up only 26% of students enrolled in the grade 10 course, 21% of students enrolled in the grade 11 courses, and 15.7% of the students enrolled in the grade 12 courses. From 2011-2018, female student enrolment in Ontario's Computer Studies has increased at a greater rate than male student enrolment, indicating that the gender gap is decreasing.

A precursory literature review shows that historically, a number of doors, walls, and windows have been identified, that inhibit certain students from equal access and participation to the computing clubhouse. Considering the number of initiatives and money related to expanding CS education, including a proposed revision of high school CS curriculum in Ontario and $80 million of additional CanCode money provided by the federal government, a better understanding of the underrepresentation of female students in high school CS is critical. Researchers within this field would be well advised to explore a number of potential theories and approaches to their work, including a technofeminist or material feminist approach, as these theoretical frameworks recognize the importance of materials in their epistemologies and they provide valuable insight into the ever-changing relationships between gender and technologies. A CS education can provide economic and educational opportunities, allow students to create and participate in a 21st century society, and help develop a diverse pool of technology talent resulting in more diverse innovations, technologies and digital solutions. Finally, as CS initiatives continue to expand to the K-8 grades, an understanding of equity, diversity, and inclusivity issues in CS is critical as new curriculum is developed. The area of curriculum within the K-8 grades will be further explored in the proceeding chapter.

4.8 Chapter References

Alaimo, S., & Hekman, S. (2008). Introduction: Emerging models of materiality in feminist theory. In S. Alaimo & S. Hekman (Eds.), *Material feminisms* (pp. 1–19). Indiana University Press.

Alphonso, C. (2021, November 11). Ontario to end streaming for all Grade 9 courses next school year. The Globe and Mail. https://www.theglobeandmail.com/canada/article-ontario-to-end-streaming-for-all-grade-9-courses-next-school-year/

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

British Columbia Ministry of Education. (2016). *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. (2018a). *Mathematics: Computer science grade 11*. https://curriculum.gov.bc.ca/sites/curriculum.gov.bc.ca/files/curriculum/mathematics/en_mathematics_11_computer-science_elab.pdf

British Columbia Ministry of Education. (2018b). *Mathematics: Computer science grade 12*. https://curriculum.gov.bc.ca/sites/curriculum.gov.bc.ca/files/curriculum/mathematics/en_mathematics_12_computer-science_elab.pdf

Cai, Z., Fan, X., & Du, J. (2017). Gender and attitudes toward technology use: A meta-analysis. *Computers & Education*, *105,* 1–13. https://doi.org/10.1016/j.compedu.2016.11.003

Canada Learning Code. (2020). *Learning for the digital world: A pan-Canadian K-12 computer science education framework*. https://k12csframework.ca/wp-content/uploads/Learning-for-the-Digital-Future_Framework_Final.pdf

Canada Learning Code. (2021). *Canada Learning Code: About us*. https://www.canadalearningcode.ca/about-us/

Ceci, S. J., Williams, W. M., & Barnett, S. M. (2009). Women's underrepresentation in science: Sociocultural and biological considerations. *Psychological Bulletin, 135*(2), 218–261. https://10.1037/a0014412

Cheryan, S., Master, A., & Meltzoff, A. N. (2015). Cultural stereotypes as gatekeepers: Increasing girls' interest in computer science and engineering by diversifying stereotypes. *Frontiers in Psychology, 6*(49), 1–8. https://doi.org/10.3389/fpsyg.2015.00049

Connelly, F. M. (2013). Joseph Schwab, curriculum, curriculum studies and educational reform. *Journal of Curriculum Studies, 45*(5), 622–639. https://doi.org/10.1080/00220272.2013.798838

Correll, S. J. (2001). Gender and the career choice process: The role of biased self-assessments. *American Journal of Sociology, 106*(6), 1691–1730. https://doi.org/10.1086/321299

Deng, Z. (2018). Contemporary curriculum theorizing: Crisis and resolution. *Journal of Curriculum Studies, 50*(6), 691–710. https://doi.org/10.1080/00220272.2018.1537376

Department of Finance Canada. (2017). *Building a strong middle class: #Budget2017*. https://www.budget.gc.ca/2017/docs/plan/budget-2017-en.pdf

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

Department of Finance Canada. (2021). *A recovery plan for jobs, growth, and resilience: Budget 2021.* https://www.budget.gc.ca/2021/home-accueil-en.html

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

Eccles, J. S., Jacobs, J. E., & Harold, R. D. (1990). Gender role stereotypes, expectancy effects, and parents' socialization of gender differences. *The Journal of Social Issues, 46*(2), 183–201. https://doi.org/10.1111/j.1540-4560.1990.tb01929.x

Ehrlinger, J., & Dunning, D. (2003). How chronic self-views influence (and potentially mislead) estimates of performance. *Journal of Personality and Social Psychology, 84*(1), 5–17. https://doi.org/10.1037/0022-3514.84.1.5

Fisher, A., & Margolis, J. (2003). Unlocking the clubhouse: The Carnegie Mellon experience. *ACM SIGCSE Bulletin, 34*(2), 79–83. https://doi.org/10.1145/543812.543836

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

http://www.edu.gov.on.ca/eng/literacynumeracy/inspire/research/Computer_Coding_K8_en.pd

Government of Canada. (2019a, March 19). *Budget 2019: Gender Equality Statement*. https://www.budget.gc.ca/2019/docs/plan/chap-05-en.html

Government of Canada. (2019b, May 6). *CanCode assessment criteria*. https://www.ic.gc.ca/eic/site/121.nsf/eng/00002.html

Government of Canada. (2019c). *CanCode*. https://www.ic.gc.ca/eic/site/121.nsf/eng/home

Government of Canada. (2020). *Funded CanCode initiatives*. https://www.ic.gc.ca/eic/site/121.nsf/eng/00003.html

Government of Canada. (2021). Canada's Digital Charter: Trust in a digital world. https://www.ic.gc.ca/eic/site/062.nsf/eng/h_00108.html

Grint, K., Gill, R., & Gill, R. M. (Eds.). (1995). *The gender-technology relation: Contemporary theory and research*. Taylor & Francis.

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

Harding, S. G. (1986). *The science question in feminism.* Cornell University Press.

Hekman, S. (2008). Constructing the ballast: An ontology for feminism. In S. Alaimo & S. Hekman (Eds.), *Material feminisms* (pp. 85–119). Indiana University Press.

Henwood, F. (2000). From the woman question in technology to the technology question in feminism: Rethinking gender equality in IT education. *European Journal of Women's Studies, 7*(2), 209–227. https://doi.org/10.1177/135050680000700209

Hmelo-Silver, C. E., & Pfeffer, M. G. (2004). Comparing expert and novice understanding of a complex system from the perspective of structures, behaviors, and functions. *Cognitive Science*, *28*(1), 127–138. https://doi.org/10.1207/s15516709cog2801_7

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

K-12 Computer Science Framework Steering Committee. (2016). *K-12 Computer Science Framework*. https://k12cs.org/

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

Kliebard, H. M. (1970). The Tyler rationale. *The School Review, 78*(2), 259–272.

Lang, C. (2010). Happenstance and compromise: A gendered analysis of students' computing degree course selection. *Computer Science Education, 20*(4), 317–345. https://doi.org/10.1080/08993408.2010.527699

Margolis, J., & Fisher, A. (2002). *Unlocking the clubhouse: Women in computing*. MIT press.

Margolis, J., Ryoo, J. J., Sandoval, C. D., Lee, C., Goode, J., & Chapman, G. (2012). Beyond access: Broadening participation in high school computer science. *ACM Inroads, 3*(4), 72-78.

Margolis, J., Estrella, R., Goode, J., Holme, J. J., & Nao, K. (2008). *Stuck in the shallow end: Education, race, and computing*. MIT press.

Margolis, J., Goode, J., & Chapman, G. (2015). An equity lens for scaling: A critical juncture for exploring computer science. *ACM Inroads*, *6*(3), 58-66.

Meadows, D. H. (2008). *Thinking in systems: A primer*. Earthscan.

Monroe, D. (2014). A New Type of Mathematics? *Communications of the ACM, 57*(2), 13 –15. https://doi.org/10.1145/2557446

Moss-Racusin, C. A., Dovidio, J. F., Brescoll, V. L., Graham, M. J., & Handelsman, J. (2012). Science faculty's subtle gender biases favor male students. *Proceedings of the National Academy of Sciences of the United States of America, 109*(41), 16474–16479. https://doi.org/10.1073/pnas.1211286109

Murphy, M. C., Steele, C. M., & Gross, J. J. (2007). Signaling threat: How situational cues affect women in math, science, and engineering settings. *Psychological Science, 18*(10), 879–885. https://doi.org/10.1111/j.1467-9280.2007.01995.x

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/Technology Vocational/Middle%20School%20Technology.pdf

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

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. (2015). *What do you need to graduate from high school?* http://www.edu.gov.on.ca/extra/eng/ppm/graduate.pdf

Ontario Ministry of Education. (2019, March 15). Education that works for you - Modernizing Learning: Province modernizing learning. *Ontario Newsroom*. https://news.ontario.ca/en/backgrounder/51527/education-that-works-for-you-modernizing-learning

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

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

Patitsas, E., Craig, M., & Easterbrook, S. (2014). A historical examination of the social factors affecting female participation in computing. In *Proceedings of the 2014 conference on innovation & technology in computer science education* (pp. 111-116). ACM.

Patitsas, E., Craig, M., & Easterbrook, S. (2015). Scaling up Women in Computing Initiatives: What Can We Learn from a Public Policy Perspective? In *Proceedings of the eleventh annual International Conference on International Computing Education Research* (pp. 61-69). ACM. https://doi.org/10.1145/2591708.2591731

Pichette, J., Deller, F., & Colyar, J. (2020) Destreaming in Ontario: History, evidence and educator reflections. Toronto: Higher Education Quality Council of Ontario. https://heqco.ca/wp-content/uploads/2020/10/Destreaming-in-Ontario_FORMATTED.pdf

Pinar, W. (Ed.). (1975). *Curriculum theorizing: The reconceptualists*. McCutchan Publishing Corporation.

Pinar, W. F., Reynolds, W. M., Slattery, P., & Taubman, P. M. (1995a). Understanding curriculum as gender text. In W. F. Pinar, W. M. Slattery, & P. M. Taubman (Eds.), *Understanding curriculum: An introduction to the study of historical and contemporary curriculum discourses* (pp. 358-403). Peter Lang Publishing.

Pinar, W. F., Reynolds, W. M., Slattery, P., & Taubman, P. M. (1995b). Understanding curriculum as a historical text: The reconceptualization of the field 1970-1979. In

W. F. Pinar, W. M. Slattery, & P. M. Taubman (Eds.*), Understanding curriculum: An introduction to the study of historical and contemporary curriculum discourses* (pp. 69-123). Peter Lang Publishing, Inc.

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

The Royal Society. (2012). *Shutdown or restart? The way forward for computing in UK schools*. https://royalsociety.org/-/media/education/computing-in-schools/2012-01-12-computing-in-schools.pdf

The Royal Society. (2017). *After the reboot: Computer education in schools*. https://royalsociety.org/~/media/policy/projects/computing-education/computing-education-report.pdf

Rudman, L. A. (1998). Self-promotion as a risk factor for women: The costs and benefits of counter stereotypical impression management. *Journal of Personality and Social Psychology, 74*(3), 629–645. https://doi.org/10.1037/0022-3514.74.3.629

Sadker, M., & Sadker, D. (1994). *Failing at fairness: How America's schools cheat girls*. Scribner.

Saskatchewan Ministry of Education. (2018a). Computer Science 20. https://www.edonline.sk.ca/webapps/moe-curriculum-BB5f208b6da4613/CurriculumHome?id=446

Saskatchewan Ministry of Education. (2018b). Computer Science 30. https://www.edonline.sk.ca/webapps/moe-curriculum-BB5f208b6da4613/CurriculumHome?id=444

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

Thompson, M., & Wiliam, D. (2007, April 9-13). *Tight but loose: A conceptual framework for scaling up school reforms* [Paper presentation]. American Educational Research Association, Chicago, IL.

United Kingdom Department for Education. (2013) *National curriculum in England: Computing programmes of study*. https://www.gov.uk/government/publications/national-curriculum-in-england-computing-programmes-of-study/national-curriculum-in-england-computing-programmes-of-study

Vakil, S. (2018). Ethics, identity, and political vision: Toward a justice-centered approach to equity in computer science education. *Harvard Educational Review, 88*(1), 26-52.

Vitores, A., & Gil-Juárez, A. (2016). The trouble with ‘women in computing’: A critical examination of the deployment of research on the gender gap in computer science. *Journal of Gender Studies, 25*(6), 666–680. https://doi.org/10.1080/09589236.2015.1087309

Wajcman, J. (2007). From women and technology to gendered technoscience. *Information Communication and Society, 10*(3), 287–298. https://doi.org/10.1080/13691180701409770

Wasserman, D., Durkee, T., & Wasserman, C. (2009). Strategies in suicide prevention. *Oxford University Press.*

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

The White House President Barack Obama. (2016, September 14). *Fact sheet: New progress and momentum in support of President Obama’s Computer Science for All Initiative*. https://obamawhitehouse.archives.gov/the-press-office/2016/09/14/fact-sheet-new-progress-and-momentum-support-president-obamas-computer

Chapter 5

5 Coding in K-8 Curriculum

Chapters 3 and 4 of this dissertation investigated the traditional implementation of CS-related concepts and skills in K-12 education, which takes the form of optional courses at the secondary level. Here in Chapter 5, the relatively new phenomenon of integrating CS-related concepts and skills into the K-8 grades is analyzed through a comparative analysis of related provincial curriculum initiatives in Canada. First, provincial K-8 curricula that includes coding and 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.

5.1 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).

Coding in the K-8 grades has become a component of these reforms. 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.

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.

5.1.1 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 centered, 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 programs, 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 overlap with Passey's arguments, but also add important 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, 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 7 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, Santo and Ching'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.

Table 7. 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 maths 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

5.1.2 Theoretical Perspectives on Coding in the K-8 Grades

When investigating theoretical approaches to coding in the younger grades, one often begins with the work of Seymour Papert, who developed the Logo programming language and the learning theory of constructionism in the 1980s. More recently, a number of theoretical approaches have been developed including Computational Thinking (Wing, 2006; Grover & Pea, 2013; Grover and Pea, 2018), Fluency (Resnick, 2018), Participation (Kafai, 2016), Literacy (diSessa, 2000, diSessa, 2018), and Action (Tissenbaum et al., 2019). In combination with the arguments for coding in the K-12 grades (listed above in Table 7), an understanding of the similarities and differences of these theoretical approaches is important in order to inform analysis of coding curricula.

5.1.2.1 Constructionism

Harel & 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. Constructionism arose from the work of Jean Piaget, with whom Papert had worked, and who articulated the theory of cognitive development called Constructivism. 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). Speaking specifically about mathematics education, the authors indicate that having students work with “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). 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).

5.1.2.2 Computational Thinking

Jeanette Wing's 2006 article, titled simply Computational Thinking, defined 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 artifacts. Her article was a call for the inclusion of CT not only in post-secondary programs 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 February, 2022 this article has been cited 8893 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.

5.1.2.3 Computational Fluency

Mirroring and expanding upon Papert's work at MIT and his development of the Logo programming language, Mitch Resnick is the director of the Lifelong Kindergarten research group that developed Scratch, currently the world's leading coding platform for kids. 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 8, 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 (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 artifacts and projects (Resnick, 2018). This development of artifacts 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.

Table 8. 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.	

5.1.2.4 Computational Participation

Sharing Resnick’s belief in the importance of students moving beyond coding puzzles to creating their own artifacts, Kafai 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 artifacts 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.

5.1.2.5 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 artifact 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.

5.1.2.6 Computational Literacy

Before Wing (2006), diSessa published his book 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 artifact 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 artifact. In Computer Coding in the K–8 Mathematics Curriculum?, Gadanidis et al. (2017) 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., 2017; Gadanidis et al., 2019), where students use or build computational models and simulations in order to better understand mathematical, scientific and engineered systems. Wilkerson & 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.

5.2 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) analyzed 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 programs and resources related to CS.

Considering these findings, this paper intends to provide 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.

5.3 Purpose and Research Questions

The purpose of this chapter 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 chapter will provide 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.

5.4 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).

5.4.1 Constructivism

This chapter 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.

5.4.2 Methodology and Document Analysis

In order to effectively answer the research questions in this study, the methodology employed will involve an initial analysis of K-8 curriculum from all Canadian provinces, with the intention of identifying where coding concepts and skills have been included. 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 this initial list of documents has been identified, a more in-depth analysis will take place involving the identification and analysis of all explicitly stated goals and rationale of this curriculum. Following this identification of curriculum, and the analysis of stated goals and objectives, document analysis will provide insight into the teaching and learning orientations of the various curricula documents.

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 analysis (Bowen, 2009). The content analysis aspect of the study will involve preliminary coding, which is the organizing of information from the documents into categories related to the central questions of the research (Bowen, 2009). This includes where explicit goals and rationale of the curriculum are identified, as well where learning outcomes or expectations are expressed.

In this study, the curriculum policy documents from Canadian provinces are 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, is conducted, identifying documents that include coding, CT and CS concepts. These documents can then be selected for content and thematic analysis, which will involve 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 is influenced by the theoretical approaches and arguments for coding presented earlier in this chapter.

5.5 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. The provinces are listed below, from West to East, as they would be presented on a map.

5.5.1 British Columbia’s Elementary Coding Curricula

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.

5.5.2 Goals of British Columbia's Elementary Coding Curricula

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 is built upon the assumption that students have a desire to create and work in practical ways.

The ADST curriculum acknowledges students as having natural curiosity, inventiveness, and a desire to create and work in practical ways. Design and creation are at the forefront of the ADST curriculum, in addition to a focus on experiential, hands-on learning, which reflects a constructionist approach to "learning by making".

A key component to the ADST curriculum is flexibility and choice, as students and teachers can personalize learning by making choices about what students "design and make, and the depth and breadth to which both teachers and students choose to pursue a particular topic, based on students' interests and passions".

5.5.3 Learning Orientations in British Columbia's Elementary Coding Curricula

The specific content and skills that students are expected to know in grades 6-8, within the Computational Thinking and Robotics modules, are listed in Table 9. 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).

Table 9. Content within the Computational Thinking and Robotics modules in British Columbia's ADST curriculum

Grade 6-7	Grade 8
Computational Thinking • simple algorithms that reflect computational thinking • visual representations of problems and data • evolution of programming languages • visual programming	**Computational Thinking** • software programs as specific and sequential instructions with algorithms that can be reliably repeated by others • debugging algorithms and programs by breaking problems down into a series of sub-problems • binary number system (1s and 0s) to represent data • programming languages, including visual programming in relation to text-based programming and programming modular component
Robotics • a robot is a machine capable of carrying out a complex series of actions automatically • uses of robotics • main components of robots: sensors, control systems, and effectors • various ways that objects can move • programming and logic for robotics components • various platforms for robotics	**Robotics** • uses of robotics in local contexts • types of sensors • user and autonomous control systems • uses and applications of end effectors • movement- and sensor-based responses • program flow • interpretation and use of schematics for assembling circuits • identification and applications of components • various platforms for robotics programming

While the stated goals and rationale highlight a belief in the very practical outcomes of coding, all of the specific concepts and skills in the CT modules for grades 6-7, and

Grade 8, do not appear to be consistent with this approach. The CT module for grade 6-7 include students knowing about the evolution of computer programming languages and perspectives, from punch cards and Ada Lovelace to Alan Turing and the Enigma machine, while in grade 8 students learn about binary number systems. This historical and computer systems design content seems better placed as context within a CS-based curriculum, rather than as CT concepts in a curriculum meant to be designed for practical and applied making and design. Neither Wing's definition and explanation of CT, nor the definition and explanation included in other CT perspectives, incorporates an understanding of the history and evolution of computer programming languages. In addition, while Wing acknowledges CT as "interpreting code as data and data as code" (Wing, 2006, p. 33), an understanding of binary systems is not recognized as a CT component that lends itself to the intended practical and applied nature of the curriculum. Considering this, it is surprising that the British Columbia curriculum includes modules identified as CT, when perhaps coding and computer programming would have been ideas that would have been a more appropriate fit, as Wing herself states that CT is conceptualizing, rather than programming, and that it is ideas, rather than artifacts.

The robotics modules, for both grades 6-7 and 8, represent the stated application based and practical nature of the curriculum goals. As an example, the historical development of robotics and automation, as well as the impact of robotics and automation and society are not included. Instead, all of the content appears to directly relate to the application and creation of robotics and automated systems. It is interesting to note that in grade 8 students will learn about robotics in local contexts, which could provide students with the opportunity to connect their learning to their world and community, conjuring images of projects that could well reflect Computational Participation and Computational Action perspectives.

5.5.4 Alberta

The elementary curriculum currently being implemented in Alberta does not include explicit coding-related concepts and skills; however, in the spring of 2021 the government released draft curriculum that included K-6 science expectations related to coding. While this is a draft document, it has been included in this analysis as it has the potential to be implemented and can provide valuable insight related to a potential direction taken to elementary coding curricula.

5.5.5 Alberta's Elementary Coding Curricula

Computational Thinking is listed as one of the major changes to Alberta's K-6 Science curricula (Alberta Education, 2021). The draft curricula website acknowledges that the old curricula did not have any references to problem solving with coding, whereas the new curricula includes "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.

5.5.6 Goals of Alberta's Elementary Coding Curricula

Computer Science has a large footprint in the draft K-8 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 curricula. 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.

Organizing ideas help structure the curriculum document, and for CS the organizing idea that spans across all grades, from K-6, is: “Problem solving and scientific inquiry are developed through the knowledgeable application of creativity, design, and computational thinking” (Alberta Education, 2021). A goal, therefore, of the coding curricula is to help students develop the ability to apply computational thinking in order to solve problems and perform scientific inquiries. Design and creativity also appear to be closely tied to the learning of CS concepts, and it appears that a major role of the coding-related concepts is to connect to, and provide context for, creativity and design.

5.5.7 Learning Orientations in Alberta’s Elementary Coding Curricula

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 10. 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.*

In addition to the guiding questions and learning outcomes, the Alberta draft curriculum document also includes Knowledge, Understanding, and Skills and Processes examples for all of the guiding questions and learning outcomes for each grade. These examples provide more depth and insight into what is expected of students. The examples provided in Kindergarten to grade 2 are interesting in that their focus is on instructions and creativity, but they are written in such a way that the use of a computer is never mentioned and potentially not necessary. In grade 3, the examples highlight CT components including breaking tasks into smaller chunks, finding patterns, and identifying important details and in grade 4, the examples highlight the theme of design by presenting a six-step design process and suggesting that students can collaborate with others to design an algorithm to solve a problem. Much like the Kindergarten to grade 2 examples, the examples in grades 3 and 4 do not use language that require a computer. It is only in grade 5 and 6 where the wording of the curriculum components make it clear that a computer must be used in order to meet the learning outcomes. A grade 5 example includes "Translate a given algorithm to block-based code" (Alberta Education, 2021, p. 49) while in grade 6 the learning outcome itself states that students will create and refine a computational artifact (Alberta Education, 2021).

Table 10. Computer Science guiding questions and learning outcomes in Alberta K-6 curriculum

	Guiding Question	Learning Outcome
Kindergarten	What are *instructions*?	Children interpret ***instructions*** in the learning environment.
Grade 1	How can we follow and create *instructions*?	Students investigate **instructions** and their influence on actions and outcomes.
Grade 2	How can *creativity* be used to ensure that *instructions* lead to the desired outcome?	Students apply ***creativity*** when designing instructions to achieve a desired outcome.
Grade 3	To what extent is *creativity* related to contributions in science?	Students investigate ***creativity*** *and* its relationship to computational thinking.
Grade 4	How can *design* resolve a problem?	Students investigate and apply ***design*** in the context of CS and technology
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	How is *design and abstraction* used in computational thinking?	Students create and refine *computational artifacts* through the use of design and ***abstraction***

5.5.8 Saskatchewan

In Saskatchewan a pilot project currently exists that is teaching robotics courses to grade 7-12 students; however, there is no formal mandated curriculum that has been implemented for all schools in the province. As a result, Saskatchewan curricula will not be a part of this analysis. Perhaps important to note; however, is that like British

Columbia, the term computational fluency, as well as the term computation, do appear in the Saskatchewan elementary mathematics curriculum. Their use does not relate to coding-related concepts and skills, and like the appearance of the term computation in the British Columbia mathematics curriculum, refers to mathematics facts and skills.

5.5.9 Manitoba

The province of Manitoba does not include any explicit, coding-related concepts and skills in any of its elementary curricula. Much like British Columbia and Saskatchewan, Manitoba's elementary mathematics curriculum discusses the importance of computational fluency and computation, but this refers to the computation taking place in the student's mind. The curriculum also includes a general learning outcome in grade 8 that reads "Approximate the square root of a number that is not a perfect square using technology (e.g., calculator, computer)" (Manitoba Education, 2013, p. 138), which would allow for coding to be used by students as an option. It is also perhaps important to note, that within the Conceptual Framework section of the curricula, technology is listed as a mathematical process. Within this section, there is a recognition that technology can be used to explore and demonstrate mathematical relationships and patterns, decrease the time spent on computations when other mathematical learning is the focus, develop personal procedures for mathematical operations, create geometric displays, and simulate situations (Manitoba Education, 2013). These are all situations in which coding may be appropriate.

5.5.10 Ontario's Elementary Coding 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 important to note, is 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). In addition to the coding expectations, the curriculum also includes one overall expectation that is related to mathematical modelling. This overall expectation is the only one in the curriculum document without accompanying specific expectations, and like coding, it is meant to be applied to various contexts within other strands. The Mathematical modelling expectation reads as follows "Overall expectation C4. apply the process of mathematical modelling to represent, analyse, make predictions, and provide insight into real-life situations" (Ontario Ministry of Education, 2020, p.4).

5.5.11 Goals of Ontario's Elementary Coding Curricula

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). The curriculum context document recognizes that technology has changed how "we access information and how students interact with mathematics" (Ontario Ministry of Education, 2020, p. 62), and an understanding that students should be able to "think critically and creatively to see connections to other disciplines beyond mathematics, such as other STEM disciplines" (Ontario Ministry of Education, 2020, p.6). 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 to experience and engage with mathematical concepts.

5.5.12 Learning Orientations in Ontario's Elementary Coding Curricula

The overall and specific expectations related to coding concepts and skills are presented in Table 11, taken directly from the Ontario curriculum document:

Table 11. Overall and specific coding expectations found in Stand C- Algebra, of the Ontario, Grades 1-8 Mathematic Curriculum

Overall Expectation C3: solve problems and create computational representations of mathematical situations using coding concepts and skills							
Specific Expectations:							
Grade 1	**Grade 2**	**Grade 3**	**Grade 4**	**Grade 5**	**Grade 6**	**Grade 7**	**Grade 8**
C3.1 solve problems and create computational representations of mathematical situations by *writing and executing* code, including code that involves *sequential events*	C3.1 solve problems and create computational representations of mathematical situations by *writing and executing* code, including code that involves sequential and *concurrent events*	C3.1 solve problems and create computational representations of mathematical situations by *writing and executing* code, including code that involves sequential, concurrent, and *repeating events*	C3.1 solve problems and create computational representations of mathematical situations by *writing and executing* code, including code that involves sequential, concurrent, repeating, and *nested events*	C3.1 solve problems and create computational representations of mathematical situations by *writing and executing* code, including code that involves *conditional statements* and other control structures	C3.1 solve problems and create computational representations of mathematical situations by *writing and executing efficient code*, including code that involves conditional statements and other control structures	C3.1 solve problems and create computational representations of mathematical situations by *writing and executing* efficient code, including code that involves events influenced by a *defined count and/or sub-program* and other control structures	C3.1 solve problems and create computational representations of mathematical situations by *writing and executing* code, including code that involves *the analysis of data in order to inform and communicate decisions*
C3.2 *read and alter* existing code, including code that involves *sequential events*, and describe how changes to the code affect the outcomes	C3.2 *read and alter* existing code, including code that involves sequential and *concurrent events*, and describe how changes to the code affect the outcomes	C3.2 *read and alter* existing code, including code that involves sequential, concurrent, and *repeating events*, and describe how changes to the code affect the outcomes	C3.2 *read and alter* existing code, including code that involves sequential, concurrent, repeating, and *nested events*, and describe how changes to the code affect the outcomes	C3.2 *read and alter* existing code, including code that involves *conditional statements* and other control structures, and describe how changes to the code affect the outcomes	C3.2 *read and alter* existing code, including code that involves conditional statements and other control structures, and describe how changes to the code affect the outcomes and the *efficiency of the code*	C3.2 *read and alter* existing code, including code that involves events influenced by a *defined count and/or sub-program* and other control structures, and describe how changes to the code affect the outcomes and the efficiency of the code	C3.2 *read and alter* existing code involving the *analysis of data in order to inform and communicate decisions*, and describe how changes to the code affect the outcomes and the efficiency of the code

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

Like the overall expectations, the specific expectations do not provide mathematical context for the problems and computational representations to be solved and created, but they do include specific coding concepts and skills. These coding concepts include, from grade 1 to grade 5, sequential, concurrent, repeating, nested and conditional events. Students from grade 1 to 5 are also expected to describe how changes to code affect outcomes. This prediction component is similar to the first step in the Predict, Run, Investigate, Modify and Make framework developed by Sentence et al. (2019). In grade 6, the concept of efficient code is added to the expectations, and in grade 7 students are asked to work with subprograms. These expectations, from 1-7, include specific coding concepts and CS concepts (control structures, subprograms, and efficiency), but they also lend themselves to CT concepts that have been discussed by Wing (2006), Brennan and Resnick (2012), and Grover and Pea (2018). Finally, in grade 8 the coding curriculum expectations refer to students using code for the analysis of data and in order to inform and communicate decisions. The specific context and source of this data is not provided, which could potentially allow teachers and students to work with data that might connect to the lives and interests of the students, or to the school and local community, which conjures images of projects that could well reflect Computational Participation and Computational Action perspectives.

5.5.13 Quebec’s Elementary Coding Curricula

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 (grade 5 and 6), as well as recognizing the impact of electric appliances, where microprocessors and computers are listed in brackets as examples (grade 3, 4, 5 and 6) (Québec Ministère de l’Éducation, 2009).

Within the Quebec mathematics curriculum, mental and written computation are also included within the arithmetic section, but like other provinces this does not refer to computation with technology.

5.5.14 Goals of Quebec’s Elementary Coding Curricula

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).

5.5.15 Learning Orientations in Quebec’s Elementary Coding Curricula

The curricula components related to students recognizing robotic structures that use servomechanisms (grade 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.

5.5.16 New Brunswick’s Elementary Coding Curricula

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 grade 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 grade 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. As previously mentioned, this document is labelled as a pilot, and therefore the scope of implementation is not yet clear, however; the document is a part of the main New Brunswick curriculum page, with a link that is found under Middle School – Technology Education.

5.5.17 Goals of New Brunswick’s Elementary Coding Curricula

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.

5.5.18 Learning Orientations in New Brunswick’s Elementary Coding Curricula

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 artifacts 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.

5.5.19 Nova Scotia’s Elementary Coding Curricula

In Nova Scotia, a key initiative of the province’s 2015 Action Plan for Education document was to “provide all students with an introduction to the basics of coding, technology, and design” (Nova Scotia Department of Education and Early Childhood Development, 2015b, p. 23). In December 2020, coding was listed as one of three main education priorities, alongside literacy and mathematics, on the province’s Education Action Plan website (Nova Scotia Department of Education and Early Childhood Development, 2021). 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).

5.5.20 Goals of Nova Scotia’s Elementary Coding Curricula

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). As previously mentioned, coding is not explicitly mentioned in formal P-3 curriculum and so these educational and economic goals are not reflected in these grades. Instead, technology operations and concepts are included, with specific reference to the safe operation of computers and grade appropriate digital devices, which reflects a more general, digital literacy goal. 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.

5.5.21 Learning Orientations in Nova Scotia’s Elementary Coding Curricula

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).

5.5.22 Prince Edward Island

The PEI Journey On documents provide specific curriculum outcomes related to communication and information technology (CIT) literacy for the K-12 grades, however; these documents do not include any formal, mandatory coding-related concepts and skills. The document explains that CIT "differs from other technologies because of its vast and far reaching applications in all disciplines" (Prince Edward Island Department of Education, 2006a, p.1), and the document highlights the importance of integrating CIT into other subject areas, rather than treating it as a subject in and of itself. Within the document there are learning outcome examples that relate to coding for webpages (hypertext mark-up language or html), but often these involve students developing a webpage using webpage development software, and then exporting the resulting design to an html version, thereby not actually coding the page itself. As an example, a grade 4 prompt suggest students can use software that will generate "required HTML coding for the layout of a particular Web page" (Prince Edward Island Department of Education, 2005a). It is also important to note that this type of HTML coding differs from computer programming coding, in that the html code is a mark-up language, rather than a programming language that is written and executed, and that automates processes.

The outcomes that do appear in the curriculum are meant to be integrated into other subject areas of the curriculum, rather than making up a stand-alone subject, and they should be used as a tool to achieve existing curricular learning outcomes within the context of other subject areas (Prince Edward Island Department of Education, 2006a). This approach is more focused on the use, rather than the development, of software tools. The document outlines the advantages of this approach which include the recognition that technology should be a tool, rather than a curriculum subject of it is own, and that the use of technology in other subject areas increase motivation and engagement, promotes the development of creative and critical thinking skills, and supports contemporary approach to education such as constructivism (Prince Edward Island Department of Education, 2006a).

5.5.23 Newfoundland and Labrador Elementary Coding Curricula

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.

5.5.24 Goals of Newfoundland and Labrador Elementary Coding Curricula

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.

5.5.25 Learning Orientations in Newfoundland and Labrador Elementary Coding Curricula

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.

5.6 Comparative Analysis and Discussion

5.6.1 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 12. A fifth category, number 2, has been added in Table 12, and while there are no provinces that make up this category, it has been added as a possible category that fits within this framework.

Category 2 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 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.

Table 12. 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 (draft) • Ontario • New Brunswick • Nova Scotia

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.

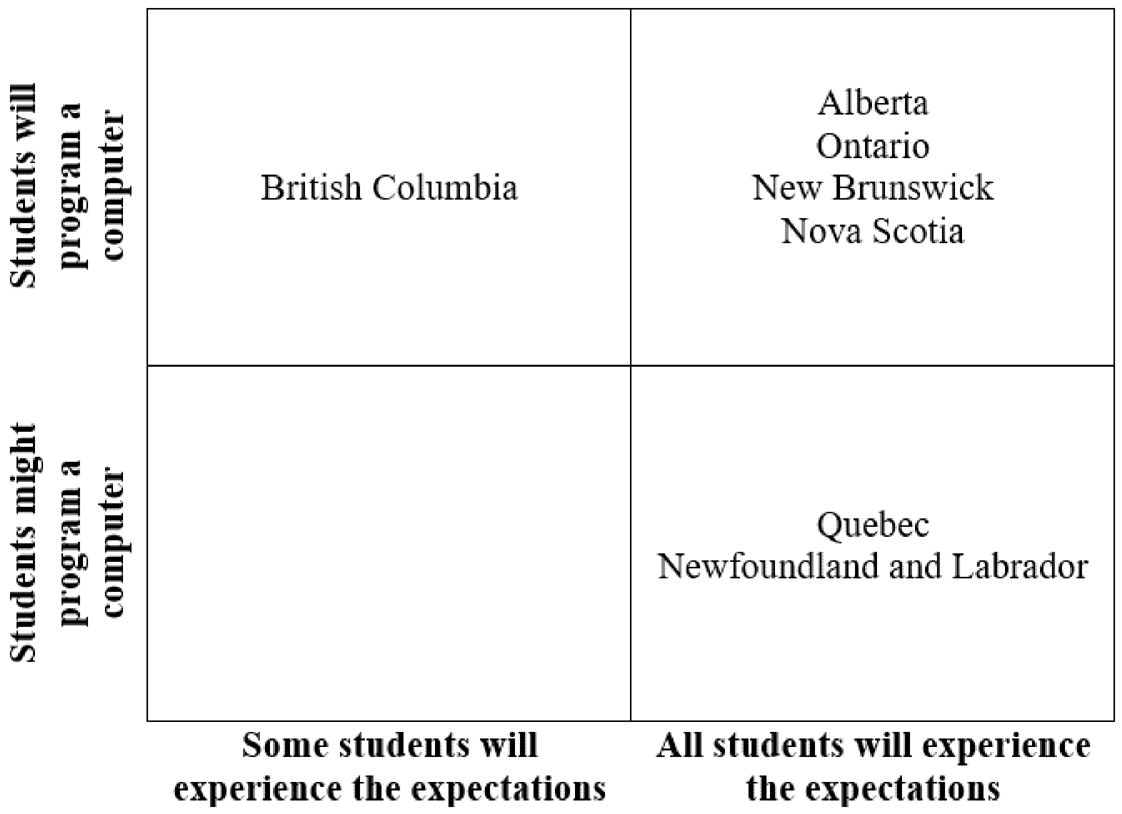
Finally, category 5 involves curriculum expectations that are written in a way that ensure that students will program a computer in order to meet the expectations, and they are found in part of the curriculum that is taught to all students. An example of this would be the expectations found in Ontario’s mathematics curriculum and the draft 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.

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 five of Table 12, 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 Figure 8.



	Some students will experience the expectations	All students will experience the expectations
Students will program a computer	British Columbia	Alberta Ontario New Brunswick Nova Scotia
Students might program a computer		Quebec Newfoundland and Labrador

Figure 8. K-8 coding curricula implementation examples from Canadian provinces

5.6.2 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:

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

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 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 draft 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 artifacts to explore physics concepts. Both these approaches recognize the value of computational artifacts to learn about and explore science concepts, yet interestingly the Alberta grade K-6 draft 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, 2016, p. 13) yet the examples do not connect the development of computational artifacts 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 artifacts 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 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 the coding work. 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.

5.6.3 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 13 lists, and briefly describes, the theoretical perspectives introduced in this chapter, 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 13 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.

Table 13. 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 • coding provides students with the opportunity to apply and extend math thinking, reasoning and communicating
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 artifacts, testing and debugging, iteration, collaboration and creativity	**BC:** • Module title is Computational Thinking • Simple algorithms that reflect CT (grade 6-7) • Visual representations of problems and data (grade 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, 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
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	**Ontario** • curriculum documents indicate that coding can be incorporated across all strands and provides students

• literacy means that a representational form for supporting intellectual activities is adopted by a broad cultural group	with opportunities to apply and extend their math thinking, reasoning, and communicating • 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 includes 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

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 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 draft coding outcomes in the K-6 Science document emphasizes a CS and CT focused approach, but does not explicitly leverage the development of computational artifacts to learn related science concepts. The Alberta draft curriculum does; however, reflect Computational Fluency, Participation, and Action, albeit with a small footprint and not as explicitly as CT.

In British Columbia ADST curricula’s reflects an emphasis on “constructionism”, and the design and creation of an artifact 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.

5.7 Conclusion

This chapter 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. 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 coding curriculum, it is clear that the economic and learning argument 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.

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 upon which to further study the novel and popular phenomenon of broadening exposure to CS-related concepts and skills.

5.8 Chapter 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

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

Ansari, A. (2017, November 21). Saskatchewan including coding in curriculum to address labour shortage. *betakit*. https://betakit.com/saskatchewan-including-coding-in-curriculum-to-address-labour-shortage/

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

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

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

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

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

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. (2017). Computer coding in the K-8 mathematics curriculum? *What works: Research into practice,* 69, 1-4.

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://10.1007/s40751-016-0019-3

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.

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. (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

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://10.5040/9781350057142.ch-003

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

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://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

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

Manitoba Education. (2013). Kindergarten to Grade 8 Mathematics: Manitoba Curriculum Framework of Outcomes. https://www.edu.gov.mb.ca/k12/cur/math/framework_k-8/full_doc.pdf

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/Technology Vocational/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. (2015a). *Information and communication technology – Essential learning outcomes 2015-2016*. https://www.ednet.ns.ca/files/curriculum/ITC-P-3ProgressionChart-RevAug26-2015.pdf

Nova Scotia Department of Education and Early Childhood Development. (2015b). *The 3Rs: Renew, refocus, rebuild - Nova Scotia's action plan for education.* https://www.ednet.ns.ca/docs/educationactionplan2015en.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

Nova Scotia Department of Education and Early Childhood Development. (2021). *Nova Scotia's Education Action Plan.* https://novascotia.ca/educationactionplan/

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

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

Prince Edward Island Department of Education. (2005a). Journey on: Working toward communication and information technology literacy grade 4. https://www.princeedwardisland.ca/sites/default/files/publications/eelc_comm_it_4.pdf

Prince Edward Island Department of Education. (2005b). Journey on: Working toward communication and information technology literacy grade 7. https://www.princeedwardisland.ca/sites/default/files/publications/eelc_comm_it_7.pdf

Prince Edward Island Department of Education. (2006a). Journey on: Working toward communication and information technology literacy grade 1. https://www.princeedwardisland.ca/sites/default/files/publications/eelc_comm_it_1.pdf

Prince Edward Island Department of Education. (2006b). Journey on: Working toward communication and information technology literacy grade 2. https://www.princeedwardisland.ca/sites/default/files/publications/eelc_comm_it_2.pdf

Prince Edward Island Department of Education. (2006c). Journey on: Working toward communication and information technology literacy grade 5. https://www.princeedwardisland.ca/sites/default/files/publications/eelc_comm_it_5.pdf

Prince Edward Island Department of Education. (2006d). Journey on: Working toward communication and information technology literacy grade 8. https://www.princeedwardisland.ca/sites/default/files/publications/eelc_comm_it_8.pdf

Prince Edward Island Department of Education. (2007a). Journey on: Working toward communication and information technology literacy grade 3. https://www.princeedwardisland.ca/sites/default/files/publications/eelc_comm_it_3.pdf

Prince Edward Island Department of Education. (2007b). Journey on: Working toward communication and information technology literacy grade 6. https://www.princeedwardisland.ca/sites/default/files/publications/eelc_comm_it_6.pdf

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

Sentance, S., Waite, J., & Kallia, M. (2019). Teaching computer programming with PRIMM: a sociocultural perspective. *Computer Science Education*, *29*(2-3), 136-176. https://doi.org/10.1080/08993408.2019.1608781

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

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. 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.

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

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. & 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

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

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

Chapter 6

6 Integrative Chapter

This integrated article dissertation set out to answer the following research question:

> What is the current, and potentially future, direction of Computer Science (CS) in K-12 education?

While this may be a broad focus, chapters 2 to 5 decomposed this large scope into specific sections that answered the following sub-questions:

a. What are the theoretical approaches presented in the literature that relate to the integration of CS concepts and skills in the K-12 grades?

b. What do curriculum documents reveal about the nature of historical CS K-12 education in terms of goals, rationale and implementation models?

c. What do enrolment patterns reveal about the nature of historical CS K-12 education in terms of equity, diversity, and inclusivity?

d. What are the CS-related concepts and skills currently found in Canadian, K-8 provincial curricula and in what ways do these reflect theoretical perspectives and historical CS K-12 education goals and rationale?

This final chapter connects the answers to these sub-questions, indicating how they intersect, and provides insights into the current and future landscape of CS in K-12 education.

6.1 Overview of Chapters 2 to 5

In Chapter 2, a review of the theoretical approaches in the field was provided. Wing's (2006) operational Computational Thinking (CT), that aims to help students make problems computable (Barba, 2016), was contrasted with other perspectives that view CS-related concepts and skills as tools that can be used to learn concepts and skills within other domains, such as mathematics and science (Papert, 1993; diSessa, 2000, 2018;

Barba, 2016), and as a means of supporting students in social, personal, and cultural endeavours (Kafai, 2016; Resnick, 2017). This analysis of the theoretical perspectives provided a foundation upon which to investigate themes in the chapters that followed.

Chapter 3 analyzed the historical CS curriculum implementation model, where CS concepts and skills have been taught in Ontario within optional courses at the secondary level since 1966. This analysis showed that many of the themes included in the preambles of recent curriculum have also been included in historical curriculum, including curriculum developed over 50 years ago. The chapter also revealed that in Ontario, historical documents have acknowledged the importance of cross-curricular connections in CS curricula and the CS courses themselves have been placed in Business, Computer Studies, Computer Science, Informatics, and Technological Education, while elsewhere we see secondary CS courses being placed in Mathematics and Science curriculum documents. With an understanding of the curriculum offered through the optional, secondary course model of implementation, it was important to consider recent enrolment within these courses.

Chapter 4 analyzed secondary, CS course enrolment data from Ontario, indicating that student enrolment has increased since the 2011-2012 school year. This increase has been primarily due to an increase in enrolment in the grade 10 ICS2O course and the grade 11 and grade 12 University pathway courses. The data also revealed a gender gap in Ontario secondary Computer Studies courses, as female students make up only 26% of students enrolled in the grade 10 course, 21% of students enrolled in the grade 11 courses, and 15.7% of the students enrolled in the grade 12 courses. From 2011-2018, female student enrolment in Ontario’s Computer Studies has increased at a greater rate than male student enrolment, indicating that the gender gap is decreasing as female student enrolment has increased by 76% while the enrolment of male students in Computer Studies courses, during that same time frame, has increased by 34%.

Finally, after considering theoretical perspectives, historical and contemporary curriculum within the optional, secondary CS courses, as well as the accompanying enrolment patterns, Chapter 5 provided evidence that the implementation of CS-related concepts and skills is expanding, beyond the optional secondary CS class and into the elementary grades in a variety of ways. Through document analysis, we can conclude 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 appeared that ranged from jurisdictions with no expectations and outcomes, to those with expectations or outcomes that guarantee coding experiences for all students. In terms of the goals of coding curriculum, it was clear that coding in order to develop job skills, as well as using coding as a tool to learn about concepts and skills related to mathematics, science, or design, were most reflected in the curriculum from the various provinces. Alternatively, perspectives related to the benefits of coding as a social practice, which might include collaboration and sharing of projects, as highlighted in Kafai's (2016) Computational Participation, were not well reflected. Learning orientations were often focused on components related to Grover and Pea's (2018) Computational Thinking concepts and practices, including abstraction, algorithms, debugging, and decomposition, while the term Computational Thinking itself appeared as a module name in BC, and was referred to in curriculum in Alberta and Nova Scotia. Computational Fluency, Computational Participation and Computational Action were not explicitly mentioned in curriculum documents. A unique integration of coding related concepts appeared in Ontario, where grades 1-8 and grade 9 Mathematics curriculum reflect components of diSessa's (2018) Computational Literacy, which will be discussed further below.

In the current chapter, the findings from these separate sections are integrated, connecting the themes uncovered in this thesis with potential future directions for K-12 CS education.

6.2 Broadening CS Education Beyond the Optional, Secondary Courses

Chapter 3 shows that there is a history of CS-related concepts and skills being implemented as isolated and optional secondary courses. A more recent trend, revealed through the document analysis performed in Chapter 5, involves jurisdictions integrating CS-related concepts and skills into subject areas other than CS, and into the elementary grades. This contemporary implementation allows for the broadening of CS education, as more students could potentially be exposed to the CS concepts and skills, even those who would not normally select to enroll in the secondary, optional CS courses.

Chapter 4 revealed a significant gender gap in the isolated and optional secondary CS courses. While the chapter did not conclude the cause of this specific gender gap, research indicates a number of doors, walls and windows that contribute to the marginalization of female students from the computing clubhouse (Margolis & Fisher, 2002). By integrating CS-related concepts and skills into learning in the elementary grades, or into other mandatory subjects in the secondary grades, jurisdictions can broaden the reach of CS concepts and skills as they will no longer only be taught to those who select the optional, secondary CS course, and will now become something that is potentially taught to all students. However, it is important to recognize the categories of implementation that were identified in Chapter 5, as expectations may be integrated as either optional or mandatory courses, and as either requiring a computer or not.

The analysis of jurisdictional coding curriculum from across Canada in Chapter 5 indicates a continuum that exists in terms of whether or not all students will be exposed to CS-related learning (see Table 14), and whether or not the learning guarantees that students will be programming a computer (see Figure 9).

The distinctions between the categories in Table 14 and Figure 9 are important when one considers implementing CS-related expectations within the elementary grades, or in other mandatory subjects in secondary, as an equity, diversity, and inclusivity initiative. If a jurisdiction integrates coding expectations in a way that does not guarantee that all students will experience the expectations, or in a way that does not guarantee explicit programming activities on a computer, then it means that a great number of students may still have to wait until secondary school to be exposed to CS-related concepts and skills, and will then only be exposed to these if they select the optional CS courses.

Table 14. Categories of implementation of coding expectations in Canadian K-8 curricula from Chapter 5

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 (draft) • Ontario • New Brunswick • Nova Scotia

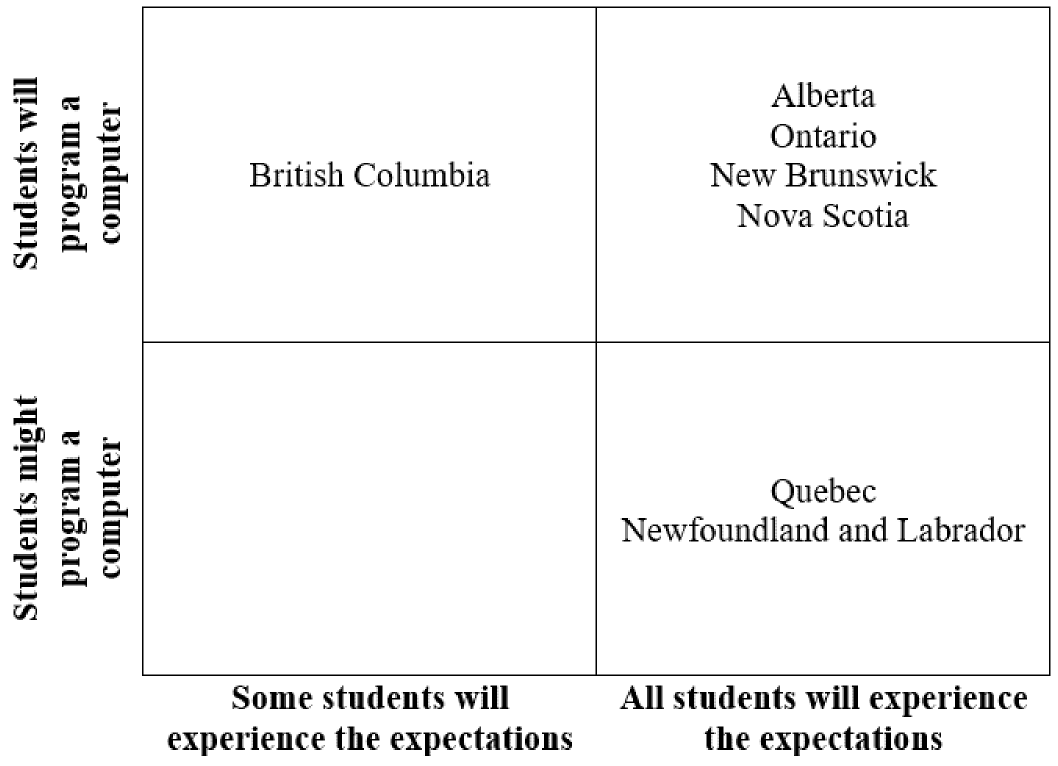

Students will program a computer
Students might program a computer
British Columbia
Alberta
Ontario
New Brunswick
Nova Scotia
Quebec
Newfoundland and Labrador
Some students will experience the expectations
All students will experience the expectations

Figure 9. K-8 coding curricula implementation examples from Canadian provinces from Chapter 5

The distinctions between these categories are also important for curriculum planning. As a result of the way that CS-related concepts and skills have been integrated into Ontario, New Brunswick, Nova Scotia, and potentially Alberta if the draft curriculum is approved, other course curriculum may be written to build upon the concepts learned in the younger grades. Other jurisdictions will not be able to do this, as their CS-related expectations are integrated as optional and can be potentially implemented without computers, and therefore only some students will have experienced these expectations through actual coding on a computer.

As more jurisdictions integrate CS-related expectations in the elementary grades or in other subject areas in high school, the categories of implementation from Chapter 5 should be carefully considered if a goal is to maximize the equity, diversity, and inclusivity benefits, and as curriculum is developed in other courses.

6.3 Papert and the Integration of CS in Other Subjects

Chapter 2 revealed a number of theoretical perspectives related to CS concepts and skills in K-12 education. Perspectives from diSessa (2000, 2018), Barba (2016), Kafai (2016), Resnick (2017), Wilkerson (Wilkerson & Fenwick, 2017) and Gadanidis (Gadanidis et al., 2019) all acknowledge and reflect Papert's Constructionism and the idea of the computer as a tool, for either understanding a specific subject (diSessa, Barba, Wilkerson, Gadanidis) or for social, personal, or cultural endeavours (Resnick and Kafai). Alternatively, Wing's (2006) CT stood apart in terms of how it did not acknowledge Papert's previous work, and how it did not reflect Papert's Constructionism. Instead, Wing's (2006) CT viewed CS and the computer as a topic for study in and of itself.

Considering the findings from Chapter 5 and the current trend of CS-related concepts and skills being integrated in curriculum from outside the isolated, high school CS discipline, perspectives that embody Papert's Constructionism and the idea of a computer as a "tool to make and do something with" are perhaps best suited as a foundation for the broadening of CS concepts and skills, and the implementation of these things in other subject areas. Papert's perspective acknowledges the computer as a tool to think with, and treats coding as something that can change the way students learn about other things.

This perspective is appropriate when CS concepts and skills are integrated into other subject areas, including mathematics and science, as the coding and computer work should serve the discipline's concepts and skills, rather than being an object of study in itself. It is also appropriate considering how CS-related concepts and skills are used within specific fields, where the focus is on the specific fields themselves, with coding as a tool to investigate, model and make progress. Having students engage with CS concepts and skills to support the learning of mathematics, or science, is appropriate considering CS concepts and skills are used within these disciplines in real life. This integrated approach therefore reflects a real-world application of CS concepts and skills, and supports students beyond simply CS specific educational and career pathways.

For examples of this effective integration, one could look to Ontario grades 1-8 and grade 9 mathematics curricula, as Chapter 5 findings show that within these curriculum documents, technology is recognized as having changed how students can interact with mathematics, and coding is recognized as providing students with the opportunity to apply and extend math thinking, reasoning, and communicating. In addition, Chapter 5 reveals that the coding expectations are written in a way that supports the learning of other mathematical concepts found in the curriculum. This implementation of coding, to support the learning of non-CS concepts and skills, is consistent with Papert's views, and the views of others presented in Chapter 2 including diSessa, Kafai, Barba, Wilkerson and Gadanidis.

In Mindstorms (1993), Papert discusses Mathland, as well as provinces of Mathland that he calls microworlds. He claims that these metaphorical places are where "certain kinds of mathematical thinking could hatch and grow with particular ease" (p. 125), much like an individual learning French by being embedded in the language while living in France. He suggests a Mathland or microworlds could be developed to serve as incubators of something like Newtonian physics, that has its own rules and structures. Students can inhabit the Newtonian physics microworld by programming various events and simulations, and these events and simulations would be impacted by the rules and structures of the particular microworld. In this way, the Newtonian physics microworld serves as a “growing place for a specific species of powerful ideas or intellectual structures” (p. 125).

With the integration of coding-related expectations in subject like mathematics in Ontario grades 1-9, the potential for Mathland or microworld experiences in schools is facilitated and supported, allowing for these intellectual incubators to be potentially experienced by a large number of students. The curriculum itself embeds coding expectations within the mathematics curriculum, and expects educators to connect these CS-related concepts and skills to mathematics concepts in the grade, effectively facilitating the learning environments that represent Mathland or microworlds for students. Papert claims that these learning experiences can reflect Piagetian learning, as they allow for learning to be deeply embedded in other activities. This is in contract to dissociated learning, which Papert claims is a symptom of mathematics learning, whereby “learning takes place in relative separation from other kinds of activities” (p. 48). This integration of coding in the curriculum, therefore, is more than simply an addition of concepts. It is instead, potentially paving the way for the development of Mathland and microworlds throughout the province in elementary mathematics classrooms, effectively facilitating integrated, rather than dissociated, experiences for students.

While CS-related curriculum may embody Papert’s approach of coding to change the way students learn about other things, resources and activities should reflect this integrated approach. In Ontario, it will be fascinating to see whether or not, over the coming years, resources and classroom activities are focused primarily on the dissociated CS concepts, or if they support the learning of associated mathematics concepts and skills. While this thesis did not investigate the tools being used or the implementation activities, considering the number of jurisdictions with new CS-related concepts and skills in their curriculum, this could serve as an interesting and necessary area of study.

It will also be fascinating to see if the integration of coding concepts in this way impacts the approaches that both educators and student take towards knowledge, learning, and potentially school in general. In discussing Mathlands, Papert claimed that “Mathland is the first step in a larger argument about how the computer presence can change not only the way we teach children mathematics, but, much more fundamentally, the way in which our culture as a whole thinks about knowledge and learning" (p.39). This begs the question whether or not the integration of coding can extend beyond reinventing mathematics activities, towards changing larger perspectives of school itself, and potentially using the computer in a wide variety of subject areas in a fashion that moves away from dissociated learning, towards richer connections and mutual support between different subject areas or concepts and skills.

As more jurisdictions integrate CS-related expectations in the elementary grades or in other subject areas in secondary school curriculum, careful consideration should be made to the theoretical perspectives from Chapter 3 and the goals of the curriculum revision initiative. Some theoretical perspectives support a CS-centric approach, and those such as Wing's CT may be most appropriate for the CS classroom, or when the computer itself is the object of study. However, if the integration of CS-related concepts and skills is meant to support the learning of, and activities within, other, non-CS subject areas, then there are other theoretical perspectives that would better serve students and better serve the goals of the curriculum.

6.4 From the Technical, to the Personal, Social and Cultural

Chapter 3 demonstrated how historical CS curriculum was focused on a technical approach, with even the most recent 2008 curriculum in Ontario stating that "Computer studies is about how computers compute. It is not about learning how to use the computer, and it is much more than computer programming. Computer studies is the study of ways of representing objects and processes" (Ontario Ministry of Education, 2008, p. 3). Enrolment data presented in Chapter 4 revealed the potential implications of this approach, as the courses implemented in an optional, isolated fashion with a focus on the technical have low enrolment and are subject to large gender gaps. While Chapter 2 demonstrated that Wing's popular CT approach embodies this CS-centric approach, other theory from the field, presented in Chapter 2 and 4, provide alternative perspectives that include personal, social, and cultural goals.

Wing's (2006) CT, while certainly a computer-centric approach, focused on students learning broader CS-related concepts and skills with the intent of solving problems and understanding the world around them. diSessa (2018) challenges this idea, arguing that Wing's CT does not address how one solves problems in general, but only a very specialized group of problems, and that her version of CT draws heavily on what he refers to as the "the siren call of higher order thinking skills" (diSessa, 2018, p. 28). He explains that Wing's CT does not provide any *filters,* for what CS concepts or skills should become common knowledge, nor does it provide principles for *lift*, or how one abstracts specific concepts and skills from CS to make them useful in broader, more general contexts. Finally, diSessa states that Wing's CT is also lacking principles of *embedding*, or how "one places abstracted elements of computation thinking in the destination disciplines so as to make them important to mathematicians, physicists, or engineers" (diSessa, 2018, p. 26).

Brennan and Resnick (2012) and Grover and Pea (2018) provided much needed details and specific components in their alternative CT concepts, practices and perspectives, and these approaches provided additional opportunities to connect CS concepts to other subject areas in school. Resnick's (2018) Computational Fluency focused on personal expression and creativity, while Kafai's (2016) Computational Participation emphasizes the cultural and social significance of coding for a purpose. diSessa (2018) provides a bigger picture perspective on the issue, as he articulates his model of computation as a new literacy that will impact the core of the STEM disciplines. Finally, in Chapter 5, Computational Action (Tissenbaum et al., 2019) was added to these approaches, where dimensions of computational identity and computational empowerment are included, as a means of making computing more inclusive, motivating, and empowering.

These approaches, while varied, place the student and the student's community at the center of the learning through personal, social, and cultural connections. This is much different than a CS-centric approach that focuses on the CS concepts and skills. New curriculum meant to broaden CS participation, whether in the secondary or elementary grades, will have to reflect this focus on the personal, the social, and the cultural otherwise it will not align with contemporary perspectives in the field such as those of Resnick (2018) and Kafai (2016).

Chapter 5 reveals that this shift has begun to occur in the development of coding curriculum in the K-8 grades. In British Columbia, coding curriculum is placed within an applied design context, where a key component is flexibility and choice, as students and teachers can personalize learning by making choices about what students “design and make, and the depth and breadth to which both teachers and students choose to pursue a particular topic, based on students’ interests and passions” (British Columbia Ministry of Education, 2016). In Alberta, a CS-centric CT is embodied within the new draft Science curriculum but some of the learning outcomes highlight creativity as a key component of intended goals (Alberta Education, 2021). Finally, in Ontario, the grade 1-8 mathematics curriculum focuses on leveraging coding to understand mathematical concepts. While specific coding-related concepts such as sequential and repeating events were included in the expectations, the focus was on solving problems and creating computational representations of mathematical situations (Ontario Ministry of Education, 2020). This is a good example of a curriculum placing the student’s personal understanding of broader learning at the forefront of coding expectations.

6.5 From the Ethical to the Justice-Centered Curriculum

Chapter 4 revealed a gender gap in high school CS courses and introduced the perspective of a justice-centered approach to equity, diversity, and inclusivity in CS education. A justice-centered approach focuses on “the sociopolitical implications, relevance, and, ultimately, liberatory possibilities of teaching and learning CS” (Vakil, 2018, p. 27). While chapter 3 provided evidence that Ontario secondary curriculum dating back as far as 1966 effectively communicated the importance of ethical considerations and digital citizenship, a justice-centered approach moves beyond developing responsible digital citizens, to students engaging in critiquing unethical abuses of technological power. In a justice-centered approach, CS learning is framed as being “important for the social and economic welfare of historically nondominant students and their communities”, as students are encouraged to “pursue CS as part of and connected to larger struggles for justice and liberation" (p. 37). New K-8 coding curriculum from various provinces explored in chapter 5 did not include this justice-centered approach, but it is worth noting that within Ontario's grade 9 mathematics curriculum, a Human Rights, Equity, and Inclusive Education in Mathematics section appears in the front matter, or curriculum context. This section includes the following paragraph which approaches some of the sociopolitical issues related to a justice-centered approach:

> Research indicates that there are groups of students (for example, Indigenous students, Black students, students experiencing homelessness, students living in poverty, students with LGBTQ+ identities, and students with special education needs and disabilities) who continue to experience systemic barriers to accessing high-level instruction in and support with learning mathematics. Systemic barriers, such as racism, implicit bias, and other forms of discrimination, can result in inequitable academic and life outcomes, such as low confidence in one's ability to learn mathematics, reduced rates of credit completion, and leaving the secondary school system prior to earning a diploma. Achieving equitable outcomes in mathematics for all students requires educators to be aware of and identify these barriers, as well as the ways in which they can overlap and intersect, which can compound their effect on student well-being, student success, and students' experiences in the classroom and in the school. Educators must not only know about these barriers, they must work actively and with urgency to address and remove them. (Ontario Ministry of Education, 2021, para. 16)

While the expectations of the curriculum do not include similar language, it is important to note that this section is included to inform educators as they deliver the grade 9 mathematics course that includes coding expectations. It is also important to note that the section did include language related to an anti-racist and decolonial approach to mathematics education, but this was removed from the section after release. The deleted text included the following:

> …mathematics has been used to normalize racism and marginalization of non-Eurocentric mathematical knowledges, and a decolonial, anti-racist approach to mathematics education makes visible its historical roots and social constructions. (Jones, 2021, para. 3)

Admittedly, this thesis does not explore the rationale for removal of the text, nor does it explore the implications of including or removing wording that acknowledges sociopolitical contexts and who has and does not have power in CS-related education and the field. This thesis does, however; acknowledge the importance of this area of study and provides a foundational analysis of what is and is not included in both historical and novel CS-related curriculum, as well as important equity, diversity, and inclusivity concerns surrounding existing gender gaps in CS education.

6.6 The Future of Secondary CS Curricula

Chapter 5 provided evidence of how the implementation of CS education is being broadened into the K-8 grades and into subject areas such as mathematics, science, and technology. Considering this phenomena, it is important to ask what the impact of this expansion will be on the secondary, optional CS courses that were explored in chapters 3 and 4.

As more students are exposed to CS concepts and skills in the younger grades, will they be motivated to enroll in CS 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 and possibly in secondary courses outside of CS, such as mathematics in grade 9 in Ontario, 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-12 CS education 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 from Chapter 2, some may consider having the secondary CS courses reflect a more CS-centric approach, embodying Wing's (2006) CT or embodying an economic argument for CS education, that prepares secondary students for post-secondary programs 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. Chapter 3 revealed that concepts of control structures in CS, which include the sequencing and repetition of instructions, as well as conditional statements (decisions), were included in all grade 10 courses over the last 55 years. 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 Chapter 5, 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-12 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.

6.7 Towards the Development of a Literacy

An evident shift in K-12 CS education involves the transition from a narrow perspective of students learning CS concepts and skills in order to pursue a related post-secondary program or career, to the broader perspective of CS-related concepts and skills potentially supporting the development of a new form of literacy. Chapter 2 and 5 highlighted the theoretical foundation of this perspective, through an analysis of diSessa's Computational Literacy (2018) which involves the adoption, by a broad group, or even a civilization, of a "particular infrastructural representational form for supporting intellectual activities" (diSessa, 2018, p. 4). diSessa presents four new Rs that provide detail and focus for his literacy agenda and all four of these Rs are reflected within the various findings from the chapters in this thesis and presented in Table 15, but are most evident in the integration of coding expectations in Ontario's grades 1-8 and grade 9 mathematics curriculum, discussed in Chapter 5.

Table 15. Examples of diSessa's (2018) four Rs in CS K-12 education

	diSessa's (2018) description of potential change	**Examples**
Re-mediation	• the computer has significantly altered the representational infrastructure of our civilization • dynamic and interactive representations are now easy and quick to create • any representational system is better adapted for some things than others	• Ontario's grade 1-8 Mathematics curriculum includes expectations related to CS concepts and skills that support the solving of problems and the representation of mathematical situations (Ontario, 2020) • British Columbia's secondary CS courses are found within the Mathematics course of study where communicating and representing is one of

		four main curricular competencies (British Columbia Ministry of Education, 2018a; 2018b)
Reformulation	• the computer can lead to substantial change in what, when and how we teach subject matter • an understanding of the different foundational ways of thinking within different domains will be important	• British Columbia's ADST curriculum leverages the application of coding concepts and skills to help students design and prototype (British Columbia Ministry of Education, 2016) • Ontario's grade 1-8 Mathematics indicates that coding can be incorporated across all strands and provides students with opportunities to apply and extend their math thinking, reasoning, and communicating (Ontario, 2020)
Reorganizing	• the intellectual terrain is changed • teaching and learning is altered significantly	• New K-8 curriculum that includes coding is apparent in a number of educational jurisdictions across Canada and the world • Ontario grades 1-8 Mathematics includes new learning expectations in the early grades that involve variables and inequalities, which are new, topics that can be well represented with the coding expectations within these grades (Ontario, 2020)
Revitalizing	• the ecology of learning activities is broadened • engagement, interest and equity is facilitated	• Papert's Constructionism is at the core of new curriculum as expectations and outcomes explicitly state that students will be programming a computer • Alberta draft curriculum includes computational artifact examples that range

		from medical research to automotive control and online shopping (Alberta Education, 2021)

6.7.1 Re-mediation

diSessa's re-mediation is reflected in findings from Chapter 5, where it was found that coding concepts and skills have been included in Ontario's grade 1-8 and 9 mathematics to support the representation of mathematical situations and in Chapter 3, where it was found that British Columbia's secondary CS curriculum is located within the mathematics discipline, and where communicating and representing form a major competency. This communicating and representing competency provides good examples of what diSessa (2018) terms a new representational infrastructure that can allow for cognitive simplicities. These include, in grade 11, students:

- explaining and justifying mathematical ideas and decisions in many ways;
- representing CS ideas in concrete, pictorial, symbolic, and pseudocode forms; and
- using CS and mathematical vocabulary and language to contribute to discussions in the classroom (British Columbia Ministry of Education, 2018a).

This type of integration of CS concepts and skills, as a new form of representational infrastructure, allows for other curriculum expectations, within the same mathematics or grade, to be learned in dynamic and interactive ways. As an example, in Ontario, a curriculum expectation in grade 3 that involves students creating and translating patterns that have repeating elements, movements, or operations can be combined with coding expectations where students are creating create computational representations of mathematical situations by writing and executing code that involves repeating events (Ontario Ministry of Education, 2020). In this case, the coding environment and the use of loops can re-mediate how students learn about and understand patters with repeating elements, movements, or operations.

6.7.2 Re-formulation

In terms of reformulation, Chapter 5 revealed students coding a computer to help support designing and prototyping in British Columbia's ADST curriculum, while in Ontario coding concepts and skills support the learning across all other strands in the mathematics curriculum and is meant to help students extend their math thinking, reasoning and communicating. These findings, in addition to the fact that Alberta includes draft coding-related curriculum in K-8 Science and that Ontario first included coding-related concepts and skills in Business data processing documents, reflects diSessa’s reformulation criteria that the computer leads to changes in what when and how we teach subject matter, and that there will be an understanding of the different ways of thinking within the different domains. As an example, Ontario’s mathematics curriculum includes students in grade 4 identifying and using symbols as variables in expressions and equations, an area where the coding environment and the storage of data in variables may alter when students can become familiar with such a concept, as well as potentially impact the depth of their understanding.

In contrast, Newfoundland and Labrador include coding expectations in Grade 8 Control Technology Module, as students define programming in terms of communications within control technology systems and as they describe the function of specific simple programs (Newfoundland and Labrador Department of Education, 2006). These types of expectations do not substantially change how subject matter is taught, or promote an understanding of foundational ways of thinking within different domains. The focus is on control systems and communication with coding used as a practical skill to control systems and devices.

6.7.3 Reorganization

diSessa’s reorganization of the intellectual terrain is apparent from findings in Chapter 5, which demonstrated the breadth and depth of curriculum revisions taking place related to coding. Since 2016, British Columbia, Alberta, Ontario, Nova Scotia and New Brunswick have all made curriculum revisions that include coding concepts and skills in their elementary curriculum. In some jurisdictions, the teaching and learning of various subjects have been changed in Canada as a result of these reforms, leading to a significant reorganization. This has changed who gets to do what, and when, as coding is potentially disruptive and alters the learning of other things.

A specific example of this is how in mathematics in Ontario, students in the early grades learn about inequalities, a topic that can be well represented with the coding expectations related to conditional statements (Ontario Ministry of Education, 2020). The intellectual terrain related to mathematical inequalities can be potentially altered by using the computer, and the computer programming code, as an object to think with. Inequalities can be represented and explored using conditional statements in code, and students may develop a deeper appreciation and understanding of the topic, and its importance and application.

In the near future, it will be fascinating to see just how much of an impact these coding expectations will have on the learning of mathematics in Ontario, and perhaps in other jurisdictions, as there is the potential for coding to "lower the floor" of some bigger mathematical concepts, and provide students with a representational infrastructure with which to wrestle with more sophisticated concepts. If this occurs, then it's possible that mathematics curriculum is reorganized, as some concepts once thought to belong in a specific grade, may be able to be moved to a younger grade. This reorganization reflects diSessa's description of how the intellectual terrain within the domain of uniform motion was re-mediated, from textual to algebraic reasoning, and therefore reorganized allowing high school students to access this learning through some inferences and the single, intuitive equation $d = rt$ (diSessa, 2018). Just as algebra reorganized the concept of uniform motion, coding may continue to reorganize mathematic, scientific, or other concepts.

In contrast, the implementation of coding-related expectations with a specific technology focus, such as in New Brunswick's Middle School Technology Education document, while supporting exciting areas of app development, robotics and electronics, may not lead to the reorganization of the intellectual terrain, and has the potential to have less of an impact on the learning within other domains.

6.7.4 Revitalization

In terms of the revitalizing of the learning ecology, this is reflected through the integration of Papert's Constructionism in many of the new coding curriculum from

Chapter 5. Many jurisdictions now include wording in their expectations that ensure that students will be programming a computer and in doing so, learning about related topics while having an “object to think with” (Papert, 1993). The broadening of this ecology and the engagement and interest is also evident in some of the examples provided in documents, including in the draft Alberta science curriculum, where computational artifacts related to automotive control and online shopping are included. This revitalization is also a key component of 21st century learning to which digital technologies and coding specifically, have been tied.

Before concluding the discussion on the shift towards a potential Computational Literacy, it should be pointed out that diSessa (2018) describes CL as “the adoption by a broad cultural group - perhaps an entire civilization - of a particular infrastructural representational form for supporting intellectual activities” (p. 4). Considering the broadening of CS-related concepts and skills in K-12 education presented in this thesis, including newly revised, explicit and mandatory coding curriculum in a number of jurisdictions in the K-8 grades, it is possible that this adoption by a broad cultural group could occur sooner than expected, as a significant number of Canadian K-8 students will be learning coding concepts and skills to support their learning in a number of different curriculum areas. While this thesis focused on the curriculum policy documents and not the actual implementation of coding expectations within the various classrooms, there is evidence of how diSessa’s CL is reflected in classrooms where coding is used to support learning.

6.7.5 Computational Literacy and a Post-Secondary Example

In Investigating an Approach to Integrating Computational Thinking into an Undergraduate Calculus Course, Clements (2020) analyzes the impact of integrating coding activities into a calculus course for undergraduate Life Sciences students. The study involved developing a set of mathematical coding activities to “supplement and enhance mathematical problem solving, as well as promote a richer understanding of the course content, while taking advantage of the unique affordances computational thinking can offer to enhance educational experiences” (p. 88). The goal was not to simply integrate technology into the course, but instead to enrich and transform how the mathematics in the course was done.

Through an analysis of questions and prompts that allowed students to reflect on their experiences with the activities, Clements (2020) observed three central themes: modified perceptions of mathematics, enhanced mathematics learning experiences, and unique coding affordances. Clements went on to analyze the findings through diSessa's (2018) Computational Literacy framework, and determined that calculus concepts in the courses were *re-mediated* with coding serving as a new computational representation system. The concepts, problems and processes related to investigations in the class were *reformulated,* sometimes from an algebraic to a computational representation, which required abstraction and automation, two CT concepts, and which also required "an in-depth conceptual understanding of all aspects of a problem, and a strong enough familiarity with both formulations that one can effectively translate between two representational systems" (p. 73). This *re-mediation* and *reformulation* resulted in a *reorganization* of course concepts, as "exploring calculus concepts with computer code enabled students to effectively investigate meaningful, authentic, interdisciplinary applications, which were formerly inaccessible (and thus omitted from the course) due to overwhelming, technical complexities" (p. 79). Clements (2020) concludes that the learning trajectories for students changed, and the intellectual domain of calculus was effectively *reorganized*, with students attributing this to the unique affordances of coding. Through *remediation, reformulation* and *reorganization,* Clements (2020) observed a *revitalization* of learning within the course, as students indicated that the coding activities:

- provided a fresh, modern approach to mathematical problem solving;
- made the material feel more interesting;
- increased their enjoyment of their learning;
- opened up a creative space in mathematics that they had never experienced in other problem-solving situations;
- allowed for flexibility in terms of the opportunities available to them, and the options available for problem-solving strategies;

- provided consistent and immediate feedback that helped them to shape and reinforce their;
- improved their confidence with their answers and overall conceptual understanding of the material;
- provided differentiated learning opportunities, which supported a variety of learning styles;
- allowed them to be free to experiment with the code in ways that were personally meaningful for them; and
- stimulated peer collaborations, resulting in fruitful discussions and sharing of ideas.

In addition to a *revitalization* of the learning, Clements was surprised to observe a revitalization of teaching, as the capabilities afforded by computation dramatically expanded the range of interdisciplinary applications that could be incorporated into the course, and expanded the capacity with which to investigate them. Clements also found that the mathematical material that was being taught could be more meaningfully and authentically engaged with, and the value of the material more convincingly illustrated.

As the coding-related expectations in Ontario mathematics, and in other subjects jurisdictions, is further implemented, it will be interesting to see if the re-mediation, reformulation, reorganization and revitalization observed by Clements in the undergraduate calculus course is also observed in the larger K-12 school system.

6.8 Broadening of CS Education Leading to New Actors and Influences

With the broadening of CS concepts and skills in K-12 education, it is likely that new actors, outside of publicly funded ministries of education, boards or schools, will become involved in the development of curriculum, the delivery of instruction, and the provision of resources and materials. An example of this is discussed in Chapter 4. Canada’s CanCode initiative begun in 2017 with an initial commitment from the Canadian federal government of $50 million (Department of Finance Canada, 2017). The program is listed as an action item related to Canada’s Digital Charter: Trust in a Digital World

(Government of Canada, 2021) and federal budgets from 2019 and 2021earmarked an additional $60 million (Department of Finance Canada, 2019) and $80 million (Department of Finance Canada, 2021) respectively for the program, resulting in provided or promised funding for the programming totaling $190 million. As discussed in Chapter 4, the CanCode program was developed to help provide coding and digital skills education to more young Canadians (Government of Canada, 2019c) and the government reports that in its first two years, had provided more than 800,000 K-12 students and 40,000 teachers with opportunities to learn these important skills (Government of Canada, 2019a).These numbers included 350,000 girls, over 68,000 Indigenous students, over 100,000 youth at risk, and 34,000 newcomers to Canada (Government of Canada, 2019a).

The funding model of this initiative is a good example of how actors outside of ministries of education, boards, and schools are involved in the development and provision of CS education for students in K-12 grades, and how this phenomena is likely to continue. To qualify for CanCode funding, groups must be a not-for-profit organization incorporated in Canada and must have a minimum of three years experience in the delivery of coding and digital skills programs to K-12 youth and/or teachers (Government of Canada, 2019b). While it was encouraged that the organizations deliver content that maps to provincial/territorial educational curricula, and while it was encouraged that the organizations partner with groups such as public school boards, neither of these criteria were mandatory. These distinctions are important as they signal that not-for-profit organizations, rather than public institutions, were selected to obtain the financial resources to lead CS education initiatives. An alternative approach would have been an investment into the broadening of CS education through groups such as Universities, Colleges, or K-12 Ministries of Education, school boards or schools.

With new CS-related curriculum being developed and implemented in the K-8 grades, and with CS expanding into other secondary subject areas, there will be a need for educational resources for students and professional development for teachers, as well as the potential purchasing of computers or other related technologies such as robotics or microcontroller kits. While this support can come from publicly funded school system groups, it’s also likely that actors from outside of the publicly funded school system, such as private STEM and coding organizations, will continue to play a role.

As this trend continues, it is important that all organizations involved in CS education in K-12 grades consider the specific goals and expectations within the jurisdictions they are supporting. Chapter 2 and 5 discussed a number of theoretical perspectives related to the integration of CS concepts and skills, it is important that organizations consider how these perspectives should be reflected in the activities, supports and technology provided to students and educators. Activities, educator professional development, pedagogical approaches and the equipment and software used to support curriculum related to the economic argument for coding will differ greatly from those used to help students learn mathematics or science concepts, or to support cultural and social endeavors.

In addition to ensuring that organizations consider the goals and expectations of the curriculum within the jurisdiction that they are supporting, it’s also important that the motivation for involvement in this work is carefully considered. With educational jurisdictions implementing CS-related concepts and skills that require a computer, or other technologies, it’s possible that some technology or educational organizations may become involved in supporting this learning for reasons that are beyond the education of students. Companies may want to sell computers and related components, or they may want to obtain student data that can be obtained when students sign in to tools or resource websites. While there are a number of organizations whose motivations may align with the motivations of educational jurisdictions, educators should be aware of this potential concern.

A final note related to these large-scale initiatives from outside of the public school system involves carefully considering the ways in which the success of these initiatives will be measured. As organizations develop resources and implement webinars and camps for students and educators, is it enough to count attendance at events or downloads of support documents? Should some type of follow-up occur, or some type of longer-range success criteria be established in order to determine whether or not initiatives had a lasting impact on broadening CS concepts and skills?

Chapter 5 discussed the various ways in which coding concepts and skills have been incorporated into the K-8 grades in different provinces, including how they have been integrated into different subjects. Organizations supporting students and educators should ensure that their activities, supports, and technology are closely tied to the specific implementation of coding in each jurisdiction, and the motivation for this involvement should be considered by educators. Making the optional criteria from the CanCode program mandatory, which encourages that organizations deliver content that maps to provincial/territorial educational curricula, and encourages that the organizations partner with groups such as public school boards, may be a way to facilitate this as partnerships between not-for-profit and publicly funded educational organizations can leverage the expertise that each organizations provides. In addition, success criteria for these initiatives should be carefully considered in order to ensure that students and educators are experiencing rich and impactful exposure to CS concepts and skills.

6.9 Research question answered

This integrated article dissertation set out to answer the following research question:

> What is the current, and potentially future, direction of CS in K-12 education?

Findings from the four preceding chapters reveal that CS in K-12 education is undergoing significant change. An analysis of related theoretical approaches shows that while Wing's (2006) operational CT, which aims to help students make problems computable (Barba, 2016), remains popular, other perspectives are being widely discussed and reflected in new curriculum revisions. These perspectives present CS-related concepts and skills as tools that can be used to learn concepts and skills within other domains, such as mathematics and science (Papert, 1993; diSessa, 2000, 2018; Barba, 2016), and as a means of supporting students in social, personal, and cultural endeavours (Kafai, 2016; Resnick, 2017).

In terms of implementation models, the delivery of CS concepts and skills as optional courses at the secondary level has been occurring for over 50 years, but new models are emerging. In Ontario, optional secondary CS-related courses have been placed in

Business, Computer Studies, Computer Science, Informatics, and Technology and enrolment data reveals that within this current, optional secondary course implementation model, less than 10% of students are enrolling in these courses. During the 2011-2012 school year, only 5% of Ontario secondary school students were enrolled in secondary Computer Studies courses. Since that time, enrolment has increased slightly, as this number reached 8% during the 2017-2018 school year. Enrolment data also showed a gender gap in Ontario secondary Computer Studies courses, but fortunately this gender gap is decreasing, as female student enrolment in Ontario's Computer Studies has increased at a greater rate than male student enrolment. From 2011 to 2018, female student enrolment has increased by 76% while the enrolment of male students in Computer Studies courses, during that same time frame, has increased by 34%.

An analysis of contemporary curriculum reveals the implementation of CS-related concepts and skills in curriculum is expanding, beyond the optional secondary CS courses, and into other subject areas and into the elementary grades. At the high school level, CS-related concepts and skills are expanding into mathematics and science programs, and within the elementary grades, 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 goals of CS-related curriculum, it is clear that coding in order to develop job skills, as well as using coding as a tool to learn about concepts and skills related to design, mathematics, and science were most reflected in the curriculum from the various provinces. British Columbia includes coding expectations that connect closely to design, which could support, and provide valuable contexts for, activities that embody Resnick's (2018) Computational Fluency and Kafai's (2016) Computational Participation perspectives. A unique integration of coding-related concepts appeared in Ontario, where grades 1-8 and grade 9 Mathematics curriculum reflect components of diSessa's Computational Literacy (2018).

Considering these findings, and the themes discussed within this integrated chapter, it is evident that a potential future direction of CS in K-12 education will include a continued broadening of skills and concepts, beyond the traditional secondary CS class, and into other grades and disciplines. Papert's view of using coding as a tool with which to learn about other things may well be the most appropriate theoretical perspective to support this work, as it serves to support the learning of concepts and skills in other disciplines, and it supports students learning how coding and computing is used in various fields. It will also be fascinating to continue to consider diSessa's (2018) big picture, Computational Literacy framework as CS-related concepts and skills are introduced to more and more students.

In terms of secondary CS courses themselves, in some jurisdictions students will be entering into these having had experience with CS-related concepts and skills in the K-8 grades, and these courses will therefore require careful considerations and revisions. While the historical gender gap within the courses remains a concern, it will be interesting to see if newer implementation models help to narrow this gap.

As a result of the findings from the various chapters, it is clear that curriculum and implementation initiatives involving CS-related concepts and skills in K-12 are undergoing significant change. Once delivered within optional, secondary courses, the current and potentially future direction of CS in K-12 education includes a reorganization of curriculum involving CS concepts and skills expanding into other subject areas, and into the younger grades. Concerns related to equity, diversity, and inclusivity play an important role in this broadening of CS education, as do big picture, theoretical perspectives related to Computational Thinking and of coding as a form of Computational Literacy. While a focus on the computer as an object of study and on the development of job ready skills remains, newer curriculum reveals the importance of the educational and social advantages of understanding and being able to apply CS-related concepts and skills. Together, these components present an exciting, and transformative time for CS education in the K-12 grades.

6.10 Limitations of the Research

This research provided an analysis of issues surrounding CS-related concepts and skills in K-12 education. An important limitation that should be addressed first is the potentially narrow scope of underrepresented groups presented in Chapter 4. The topic of underrepresentation in CS education, and the field itself, is critical and also complex. Chapter 4 only covers one concern, the underrepresentation of female students in secondary Computer Studies courses, and while every attempt was made to provide as much detail as possible, the complexity of such an important topic was difficult to fully represent in a single chapter. In order to assuage these concerns, an attempt was made to reference as many researchers and works as possible. A thesis on CS education would not be complete without addressing this area, which is why it was important to the author to include Chapter 4. Furthermore, it should be re-emphasized that educators, policy makers, and researchers should certainly avoid the tendency to employ a deficit approach when discussing underrepresented groups in CS education. Margolis and Fisher presented the doors, walls and windows of the computing clubhouse back in 2002, and these lessons of systemic barriers should continue to be heeded today. Additionally, the binary classification of students as female and male, in Chapter 4, was in order to stay consistent with the classifications in the data provided by the Ministry of Education.

A second limitation to the research is that the main focus of the various chapters was on the aims and goals, and expectations and outcomes, of coding curriculum policy documents rather than the implementation or pedagogy related to the curriculum, and the work being done in the classrooms. This thesis would have been strongly supported by an analysis of areas such as the coding arguments and approaches reflected when educators integrate the identified curriculum, or some type of evaluation of the success of curriculum in achieving stated aims and goals. It was felt, however; that considering the novel nature of coding expectations in the K-8 grades, and the recent explosion of interest in integrating coding in the younger grades, that an analysis of curriculum policy was critical at this stage. This study also provides insights for researchers and policy makers, as they continue to consider and develop coding curriculum that will support the important implementation stages executed by educators. It is also hoped that this study provides a foundation upon which researchers can build, in order to develop studies that provide valuable insight into implementation stages and associated pedagogy.

Finally, an acknowledged and important limitation of this research is that an assessment and evaluation lens was never used when considering the arguments and approaches for coding in the younger grades, the historical Computer Studies curriculum in Ontario, the K-8 coding curriculum from Canadian provinces, or the issues of underrepresented groups in CS education. This assessment and evaluation lens is an important one as researchers and policymakers continue to develop novel coding curriculum, and as educators continue to implement coding expectations and outcomes. As new coding curriculum is developed, researchers and policy makers need to be aware of assessment and evaluation policies and practices within their jurisdictions, to ensure that coding expectations and outcomes are appropriately written and aligned, and that educators can effectively assess and evaluate student work.

6.11 Implications and Future Research

As indicated in the limitations of research section, the findings from this study provide researchers with foundational understandings upon which to build. Chapter 2 provides scholars new to the field with a clear and cohesive description and comparison of theoretical approaches to coding, CS, and CT in K-12 education. This description and comparison of approaches and directions could form the basis, or serve as a framework, for a number of studies analyzing teacher or parent perspectives on coding or evaluating the orientations of pedagogy and classroom activities implemented by teachers.

The findings from Chapter 3, related to historical Computer Studies curriculum, provide evidence that while coding-related concepts and skills in the K-8 grades may be new, in some jurisdictions coding in secondary curriculum dates back as far as 1966. The chapter also provides evidence that many of the aims and objectives of historical curriculum are shared with modern approaches. This has significance for policy makers, as it demonstrates that historical curriculum could be a source of insight for the development of new curriculum, especially if studies are done comparing the implementation, pedagogy or classroom activities related to historical curriculum, with the implementation, pedagogy or classroom activities related to the curriculum of today. There is also potential to compare historical CS curriculum from other jurisdictions, and from post-secondary institutions, in order to evaluate the evolution and innovation, or lack of evolution and innovation, within the curriculum over the years. This could shed light on whether or not the field of CS curriculum is one that evolves and improves, or one that stagnates or remains the same.

Chapter 4 raises a number of issues related to the underrepresentation of specific groups in CS education and the field, and could serve as a starting point for a critical analysis of curriculum, pedagogy, or classroom activities in CS education. The area of equity, diversity, and inclusivity are of upmost importance in CS education, as is the recognition and acknowledgement of bias. Awareness of these concerns could serve as a positive first step, that could be followed by research that goes beyond superficialities, and instead dive into the potential systemic issues at play. In addition, culturally responsive, anti-racist, and anti-colonialist curriculum and pedagogy are areas in which research can be done to support much needed change in CS education, and in education in general.

Chapter 5's findings, related to Canadian provincial coding curriculum in the K-8 grades, can hopefully add to the research from other jurisdictions, and can serve as an additional perspective as researchers continue to investigate new approaches to coding in the younger grades. Internationally, it would be interesting to compare the aims and goals of Canadian provincial coding curriculum to those of other jurisdictions and countries. Within Canada, it would be interesting to study how the implementation of coding curriculum differs in the various provinces, while considering the different ways in which the coding curriculum was written. The five categories of curriculum integration and the three subject areas identified could also serve as the foundation for a framework with which to analyze other K-8 coding curricula.

In addition to the findings from this work, it is hoped that the methods and frameworks employed can help inform or frame future studies. Document analysis and Thematic Analysis could be useful for researchers investigating coding in K-8 education, as these lend themselves to the analysis of documents and policy that continues to be developed in the field. Chapter 4 concludes by introducing a number of frameworks and perspectives, including justice-centered CS education, technofeminism and material feminism, that if employed in the CS education context, could provide valuable and much needed perspectives in a field that is historically androcentric. These types of theoretical frameworks should be seen as powerful tools for positive change.

Throughout this work, a number of contemporary approaches and programs are identified and described. This work, therefore serves as a timestamp, identifying what exists now, at this current time, within the area of CS K-12 education. This timestamp may be of value to future researchers, as they compare new initiatives to those of the past, and as they consider the journey of CS K-12 education.

Finally, this work provides an analysis of the current state of CS K-12 education during a transformative time. It identifies new and exciting themes and programs related to the broadening of CS concepts and skills, including federal programs such as CanCode and CSForAll, as well as new coding curriculum in the K-8 grades from various provinces. At the same time, this work provides evidence that while the field of CS K-12 education is being influenced by new perspectives and programs, CS education in the K-12 grades has a past that includes research, theoretical perspectives, and curriculum. It's important that educators, policy makers, and researchers acknowledge and learn from contemporary and historical research, curriculum, and programs in order to help shape a successful future for CS K-12 education.

6.12 Conclusion

This integrated article dissertation provides an in-depth analysis of the current state of K-12 CS education through the lenses of theory in the field, historical and novel curricula, and student enrolment and equity, diversity, and inclusivity. Chapter 2 presented theoretical approaches and directions taken by leading researchers in the field, including Computational Thinking, Computational Fluency, Computational Participation, Computational Action, Computational Modeling and Computational Literacy. Chapter 3 provided evidence that while many coding initiatives in the K-8 grades are new, historical secondary CS curriculum exists, and is worth investigating as a means of supporting new curriculum initiatives. Chapter 4 analyzed enrolment data related to the isolated and optional implementation model of CS courses in secondary and confirmed a significant gender gap. This chapter also presented a vital look at initiatives, frameworks, and perspectives that could help CS educators, policy makers and researchers tackle important equity, diversity, and inclusivity concerns in the field. Finally, Chapter 5 presented various arguments for coding in the younger grades, including those related to economics, education, culture, and society. The chapter also provided an analysis of the placement, goals, and learning orientations of coding expectations and outcomes in Canadian provincial, K-8 curriculum. The categories developed through this document analysis could serve as a valuable starting point for policymakers and researchers engaging in coding and curriculum work.

While all of these findings present important pieces of the CS K-12 education puzzle, it is important to remind ourselves of the big picture. The big picture of CS education often involves a student and a computer, and the magic that can take place when these two interact. It therefore seems appropriate that a PhD dissertation related to computers, coding, and education should end with a final thought from Seymour Papert, the father of computing education (Stager, 2016). In Let's Tie the Digital Knot, Papert (1998) discusses a number of topics related to wholesale, educational reform. He points out the absurdity of the term "Computers in Education", by highlighting the fact that we do not hold conferences called "The World Congress on Paper-Based Education", and we do not publish papers in the "Journal of Computer-Free Schooling". He explains that educators, researchers, and policy makers should be technologically fluent individuals who have absorbed computational ideas into their culture, and who desire to see changes in learning that others cannot even imagine. He calls us to have "more chutzpah" in order to replace the use of technology to improve education, with a call to invent new visions of education in the context of this digital world. Such grand visions of education reform such as this are daunting, but Papert offers a prescription: "simply spend time doing it – the muscle of the mind will grow through exercise" (p. 2).

If you are reading this dissertation then I assume you are engaged in this work, you are exercising the muscles of your mind, and you are a part of a grand vision of education reform, that focusses on improving education for ALL students in this digital world. A concluding thought, therefore, which bolstered, and continues to bolster, this author through his work, and which may help the reader in theirs, is to consider the following from Papert:

> In my trademark caricature of this situation, a nineteenth-century transportation engineer invents a jet engine and attaches it to a stagecoach to assist the horses. But the transformative contribution of the jet engine to transportation did not come from improving already existing vehicles. It came through the invention of a radically new kind of vehicle - the jet plane. (Papert, 1998, p. 2)

The computer represents an engine that can provide thrust to a form of education that affords our students with new, engaging, rich, and valuable learning experiences never thought possible. It could also potentially lead to altered representational infrastructure, substantial change in what, when, and how we teach subject matter, a change in the overall intellectual terrain as teaching and learning is altered, and in the overall broadening of the ecology of learning (diSessa, 2018). In short, it could lead to a potential new literacy, but a key component to all of this is the development of effective curriculum that is appropriate for, and experienced by, all of our students.

This thesis identifies the approaches, arguments, directions, philosophies, aims, objectives, challenges, and goals related to CS education in the K-12 grades. It is meant to help educators, researchers and policy makers better understand the historical, current, and potential future state of K-12 CS education. It is also meant to help us better support students as they learn the concepts and skills needed to design, build, and pilot their own jet plane and to assume their unique and rightful place amongst the stars.

6.13 Chapter 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

Barba, L. (2016, March 15). Computational thinking: I do not think it means what you think it means. *Lorena A. Barba Group*. http://lorenabarba.com/blog/computational-thinking-i-do-not-think-it-means-what-you-think-it-means/.

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. (2016). Applied design, skills and technologies: Goals and rationale. https://curriculum.gov.bc.ca/curriculum/adst/goals-and-rationale

British Columbia Ministry of Education. (2018a). Mathematics: Computer science grade 11. https://curriculum.gov.bc.ca/sites/curriculum.gov.bc.ca/files/curriculum/mathematics/en_mathematics_11_computer-science_elab.pdf

British Columbia Ministry of Education. (2018b). Mathematics: Computer science grade 12. https://curriculum.gov.bc.ca/sites/curriculum.gov.bc.ca/files/curriculum/mathematics/en_mathematics_12_computer-science_elab.pdf

Clements, E. (2020). *Investigating an approach to integrating Computational Thinking into an undergraduate calculus course* [Doctoral dissertation, Western University]. Electronic Thesis and Dissertation Repository.

Department of Finance Canada. (2017). *Building a strong middle class: #Budget2017*. https://www.budget.gc.ca/2017/docs/plan/budget-2017-en.pdf

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

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

Government of Canada. (2019a, March 19). *Budget 2019: Gender Equality Statement*. https://www.budget.gc.ca/2019/docs/plan/chap-05-en.html

Government of Canada. (2019b, May 6). *CanCode assessment criteria*. https://www.ic.gc.ca/eic/site/121.nsf/eng/00002.html

Government of Canada. (2019c). *CanCode*. https://www.ic.gc.ca/eic/site/121.nsf/eng/home

Government of Canada. (2020). *Funded CanCode initiatives*. https://www.ic.gc.ca/eic/site/121.nsf/eng/00003.html

Government of Canada. (2021). Canada's Digital Charter: Trust in a digital world. https://www.ic.gc.ca/eic/site/062.nsf/eng/h_00108.html

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/0.5040/9781350057142.ch-003

Jones, A. (2021, July 14). Ontario removes anti-racism language from math curriculum preamble. *Toronto Star*. https://www.thestar.com/news/gta/2021/07/14/ontario-removes-anti-racism-text-from-math-curriculum-preamble.html

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

Margolis, J., & Fisher, A. (2002). *Unlocking the clubhouse: Women in computing*. MIT press.

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

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

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

Papert, S. (1998). Let’s tie the digital knot. *TECHNOS Quarterly for Education and Technology, 7*(4). http://dailypapert.com/wp-content/uploads/2018/06/Papert-Lets-Tie-the-Digital-Knot.pdf

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

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

Stager, G. S. (2016). Seymour Papert (1928-2016). *Nature, 537*(7620), 308-308. https://doi.org/10.1038/537308a

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

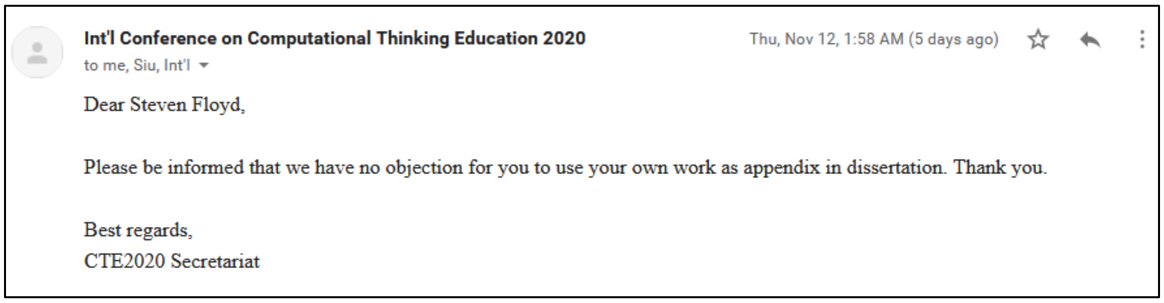
Vakil, S. (2018). Ethics, identity, and political vision: Toward a justice-centered approach to equity in computer science education. *Harvard Educational Review, 88*(1), 26-52.

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.

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

Appendices

Appendix A. Email from CTE 2020 Secretariat providing reprint permission.



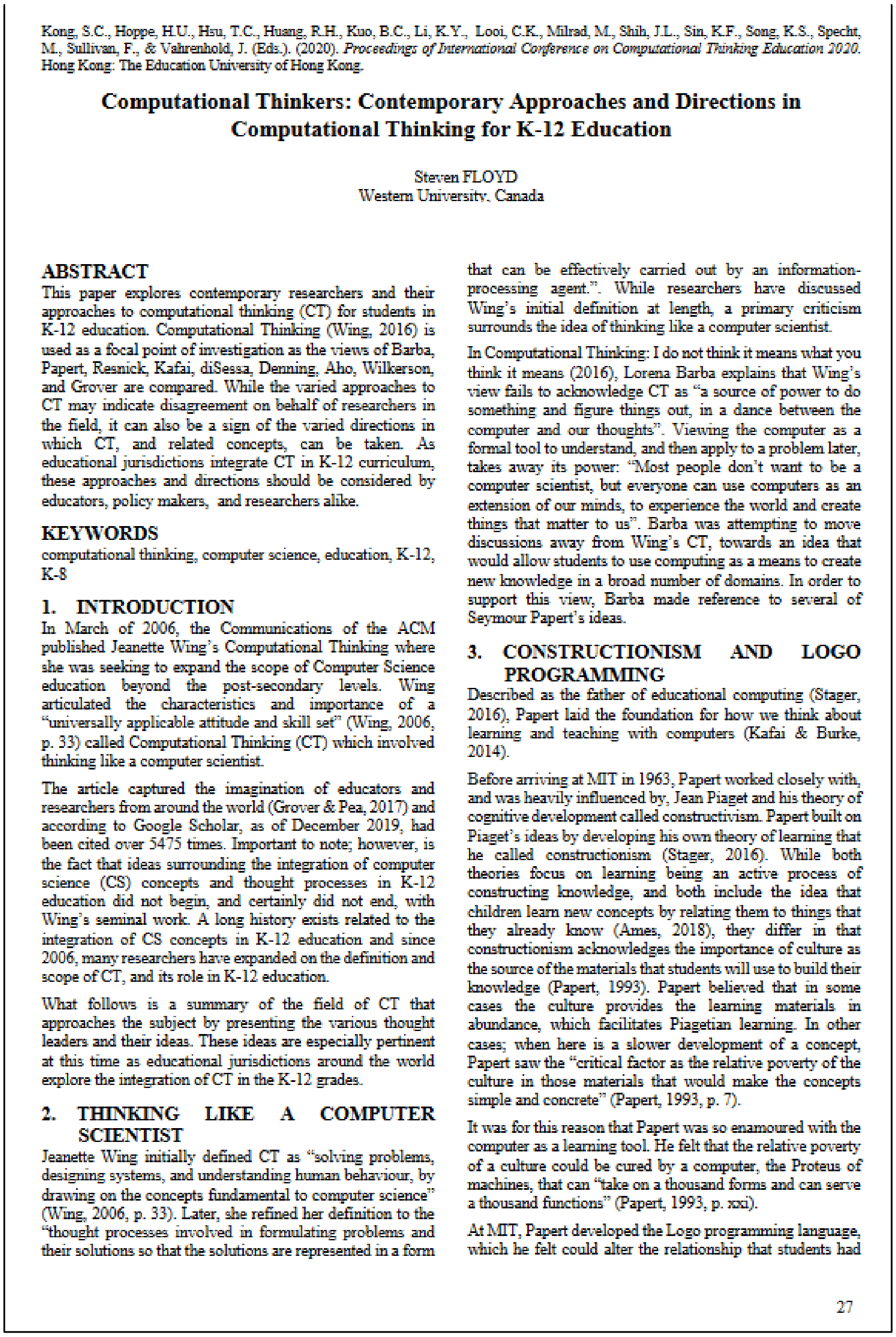
Int'l Conference on Computational Thinking Education 2020
to me, Siu, Int'l
Thu, Nov 12, 1:58 AM (5 days ago)

Dear Steven Floyd,

Please be informed that we have no objection for you to use your own work as appendix in dissertation. Thank you.

Best regards,
CTE2020 Secretariat

Appendix B. Published paper from CTE 2020.



Kong, S.C., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Shih, J.L., Sin, K.F., Song, K.S., Specht, M., Sullivan, F., & Vahrenhold, J. (Eds.). (2020). *Proceedings of International Conference on Computational Thinking Education 2020*. Hong Kong: The Education University of Hong Kong.

Computational Thinkers: Contemporary Approaches and Directions in Computational Thinking for K-12 Education

Steven FLOYD
Western University, Canada

ABSTRACT

This paper explores contemporary researchers and their approaches to computational thinking (CT) for students in K-12 education. Computational Thinking (Wing, 2016) is used as a focal point of investigation as the views of Barba, Papert, Resnick, Kafai, diSessa, Denning, Aho, Wilkerson, and Grover are compared. While the varied approaches to CT may indicate disagreement on behalf of researchers in the field, it can also be a sign of the varied directions in which CT, and related concepts, can be taken. As educational jurisdictions integrate CT in K-12 curriculum, these approaches and directions should be considered by educators, policy makers, and researchers alike.

KEYWORDS

computational thinking, computer science, education, K-12, K-8

1. INTRODUCTION

In March of 2006, the Communications of the ACM published Jeanette Wing's Computational Thinking where she was seeking to expand the scope of Computer Science education beyond the post-secondary levels. Wing articulated the characteristics and importance of a "universally applicable attitude and skill set" (Wing, 2006, p. 33) called Computational Thinking (CT) which involved thinking like a computer scientist.

The article captured the imagination of educators and researchers from around the world (Grover & Pea, 2017) and according to Google Scholar, as of December 2019, had been cited over 5475 times. Important to note; however, is the fact that ideas surrounding the integration of computer science (CS) concepts and thought processes in K-12 education did not begin, and certainly did not end, with Wing's seminal work. A long history exists related to the integration of CS concepts in K-12 education and since 2006, many researchers have expanded on the definition and scope of CT, and its role in K-12 education.

What follows is a summary of the field of CT that approaches the subject by presenting the various thought leaders and their ideas. These ideas are especially pertinent at this time as educational jurisdictions around the world explore the integration of CT in the K-12 grades.

2. THINKING LIKE A COMPUTER SCIENTIST

Jeanette Wing initially defined CT as "solving problems, designing systems, and understanding human behaviour, by drawing on the concepts fundamental to computer science" (Wing, 2006, p. 33). Later, she refined her definition to the "thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent.". While researchers have discussed Wing's initial definition at length, a primary criticism surrounds the idea of thinking like a computer scientist.

In Computational Thinking: I do not think it means what you think it means (2016), Lorena Barba explains that Wing's view fails to acknowledge CT as "a source of power to do something and figure things out, in a dance between the computer and our thoughts". Viewing the computer as a formal tool to understand, and then apply to a problem later, takes away its power: "Most people don't want to be a computer scientist, but everyone can use computers as an extension of our minds, to experience the world and create things that matter to us". Barba was attempting to move discussions away from Wing's CT, towards an idea that would allow students to use computing as a means to create new knowledge in a broad number of domains. In order to support this view, Barba made reference to several of Seymour Papert's ideas.

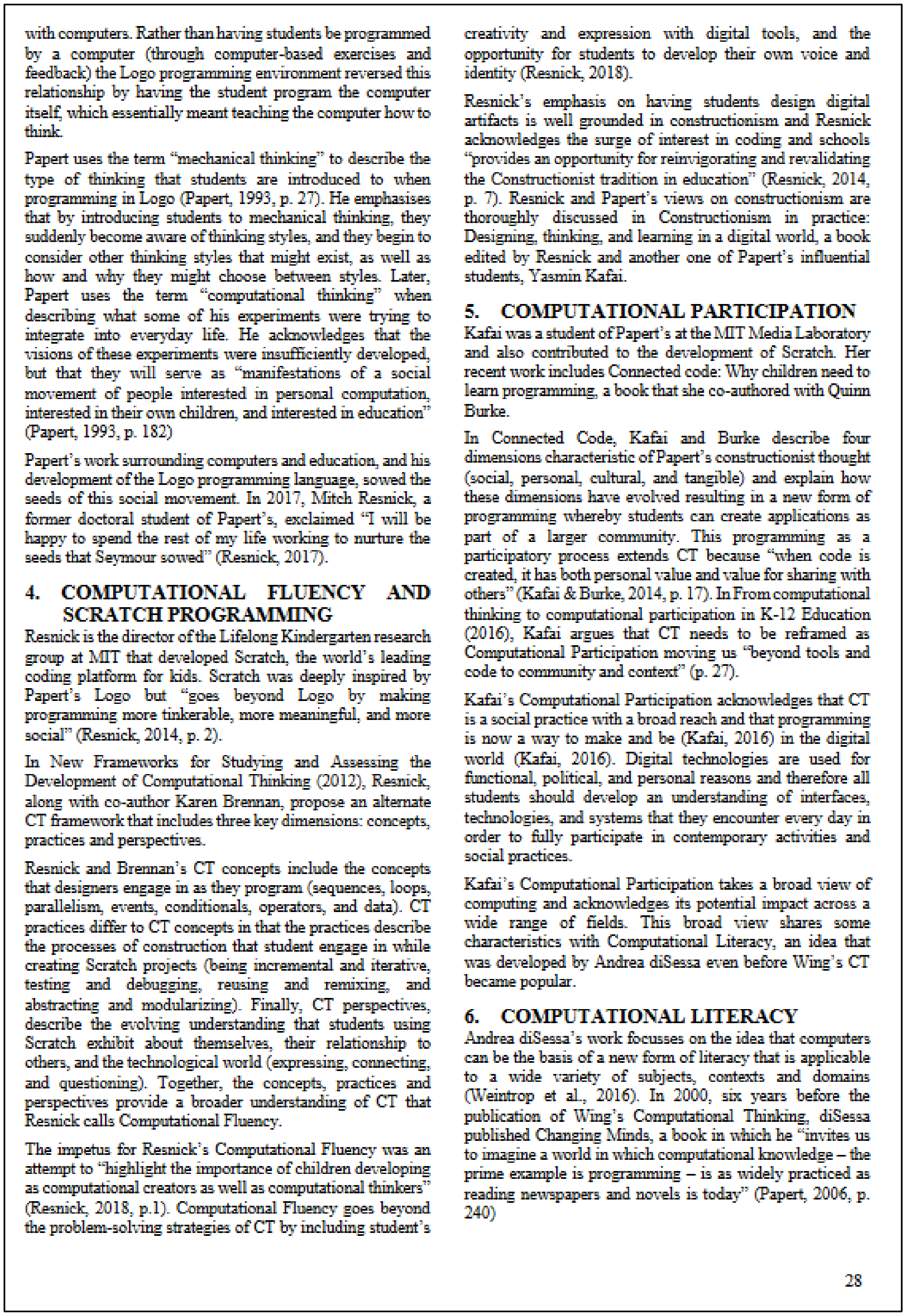
3. CONSTRUCTIONISM AND LOGO PROGRAMMING

Described as the father of educational computing (Stager, 2016), Papert laid the foundation for how we think about learning and teaching with computers (Kafai & Burke, 2014).

Before arriving at MIT in 1963, Papert worked closely with, and was heavily influenced by, Jean Piaget and his theory of cognitive development called constructivism. Papert built on Piaget's ideas by developing his own theory of learning that he called constructionism (Stager, 2016). While both theories focus on learning being an active process of constructing knowledge, and both include the idea that children learn new concepts by relating them to things that they already know (Ames, 2018), they differ in that constructionism acknowledges the importance of culture as the source of the materials that students will use to build their knowledge (Papert, 1993). Papert believed that in some cases the culture provides the learning materials in abundance, which facilitates Piagetian learning. In other cases; when here is a slower development of a concept, Papert saw the "critical factor as the relative poverty of the culture in those materials that would make the concepts simple and concrete" (Papert, 1993, p. 7).

It was for this reason that Papert was so enamoured with the computer as a learning tool. He felt that the relative poverty of a culture could be cured by a computer, the Proteus of machines, that can "take on a thousand forms and can serve a thousand functions" (Papert, 1993, p. xxi).

At MIT, Papert developed the Logo programming language, which he felt could alter the relationship that students had



with computers. Rather than having students be programmed by a computer (through computer-based exercises and feedback) the Logo programming environment reversed this relationship by having the student program the computer itself, which essentially meant teaching the computer how to think.

Papert uses the term "mechanical thinking" to describe the type of thinking that students are introduced to when programming in Logo (Papert, 1993, p. 27). He emphasises that by introducing students to mechanical thinking, they suddenly become aware of thinking styles, and they begin to consider other thinking styles that might exist, as well as how and why they might choose between styles. Later, Papert uses the term "computational thinking" when describing what some of his experiments were trying to integrate into everyday life. He acknowledges that the visions of these experiments were insufficiently developed, but that they will serve as "manifestations of a social movement of people interested in personal computation, interested in their own children, and interested in education" (Papert, 1993, p. 182)

Papert's work surrounding computers and education, and his development of the Logo programming language, sowed the seeds of this social movement. In 2017, Mitch Resnick, a former doctoral student of Papert's, exclaimed "I will be happy to spend the rest of my life working to nurture the seeds that Seymour sowed" (Resnick, 2017).

4. COMPUTATIONAL FLUENCY AND SCRATCH PROGRAMMING

Resnick is the director of the Lifelong Kindergarten research group at MIT that developed Scratch, the world's leading coding platform for kids. Scratch was deeply inspired by Papert's Logo but "goes beyond Logo by making programming more tinkerable, more meaningful, and more social" (Resnick, 2014, p. 2).

In New Frameworks for Studying and Assessing the Development of Computational Thinking (2012), Resnick, along with co-author Karen Brennan, propose an alternate CT framework that includes three key dimensions: concepts, practices and perspectives.

Resnick and Brennan's CT concepts include the concepts that designers engage in as they program (sequences, loops, parallelism, events, conditionals, operators, and data). CT practices differ to CT concepts in that the practices describe the processes of construction that student engage in while creating Scratch projects (being incremental and iterative, testing and debugging, reusing and remixing, and abstracting and modularizing). Finally, CT perspectives, describe the evolving understanding that students using Scratch exhibit about themselves, their relationship to others, and the technological world (expressing, connecting, and questioning). Together, the concepts, practices and perspectives provide a broader understanding of CT that Resnick calls Computational Fluency.

The impetus for Resnick's Computational Fluency was an attempt to "highlight the importance of children developing as computational creators as well as computational thinkers" (Resnick, 2018, p.1). Computational Fluency goes beyond the problem-solving strategies of CT by including student's creativity and expression with digital tools, and the opportunity for students to develop their own voice and identity (Resnick, 2018).

Resnick's emphasis on having students design digital artifacts is well grounded in constructionism and Resnick acknowledges the surge of interest in coding and schools "provides an opportunity for reinvigorating and revalidating the Constructionist tradition in education" (Resnick, 2014, p. 7). Resnick and Papert's views on constructionism are thoroughly discussed in Constructionism in practice: Designing, thinking, and learning in a digital world, a book edited by Resnick and another one of Papert's influential students, Yasmin Kafai.

5. COMPUTATIONAL PARTICIPATION

Kafai was a student of Papert's at the MIT Media Laboratory and also contributed to the development of Scratch. Her recent work includes Connected code: Why children need to learn programming, a book that she co-authored with Quinn Burke.

In Connected Code, Kafai and Burke describe four dimensions characteristic of Papert's constructionist thought (social, personal, cultural, and tangible) and explain how these dimensions have evolved resulting in a new form of programming whereby students can create applications as part of a larger community. This programming as a participatory process extends CT because "when code is created, it has both personal value and value for sharing with others" (Kafai & Burke, 2014, p. 17). In From computational thinking to computational participation in K-12 Education (2016), Kafai argues that CT needs to be reframed as Computational Participation moving us "beyond tools and code to community and context" (p. 27).

Kafai's Computational Participation acknowledges that CT is a social practice with a broad reach and that programming is now a way to make and be (Kafai, 2016) in the digital world (Kafai, 2016). Digital technologies are used for functional, political, and personal reasons and therefore all students should develop an understanding of interfaces, technologies, and systems that they encounter every day in order to fully participate in contemporary activities and social practices.

Kafai's Computational Participation takes a broad view of computing and acknowledges its potential impact across a wide range of fields. This broad view shares some characteristics with Computational Literacy, an idea that was developed by Andrea diSessa even before Wing's CT became popular.

6. COMPUTATIONAL LITERACY

Andrea diSessa's work focusses on the idea that computers can be the basis of a new form of literacy that is applicable to a wide variety of subjects, contexts and domains (Weintrop et al., 2016). In 2000, six years before the publication of Wing's Computational Thinking, diSessa published Changing Minds, a book in which he "invites us to imagine a world in which computational knowledge – the prime example is programming – is as widely practiced as reading newspapers and novels is today" (Papert, 2006, p. 240)

In presenting computing as a new form of literacy, diSessa advocated for the broad use of computers in schools, and for educators to see computing as means of transforming the teaching and learning of things that are hard for students to learn (Papert, 2006). diSessa uses algebra as an example of an epistemological entity that, when first developed, was not appreciated as a means of transforming complex and difficult ideas into a form that can be grasped by high school students (Papert, 2006). He argues that Computational Literacy involves computing and computer programming concepts being integrated into school subjects in much the same way that algebra has become a tool in science, mathematics and other subjects.

In Computational Literacy and "The Big Picture" Concerning Computers in Mathematics Education (2018), diSessa explains that his use of the term literacy goes beyond the idea of simply having a casual acquaintance with something. Instead, literacy means the adoption, by a broad group or even a civilization, of a "particular infrastructural representational form for supporting intellectual activities" (diSessa, 2018, p. 4). diSessa continues by criticizing Wing's computer science-centric view of CT acknowledging that because literacy is such a massive social and intellectual accomplishment, it can't belong to a single professional discipline.

diSessa concludes Computational Literacy and "The Big Picture" Concerning Computers in Mathematics Education by providing practical advice:

There is no single recipe for how computation changes a field or subfield. If your pursuits take you in different directions, then I suggest here, that will enrich the horizon for all of us. If they parallel or extend what I and others who are focused on the big picture have already done, perhaps we can converge sooner than might be expected (diSessa, 2018, p. 28).

We should consider this advice as we investigate the views and applications of CT shared by other researchers within the field.

7. CT DEFINITIONS AND MATHEMATICAL MODELS

In 2017, Peter Denning published Remaining trouble spots in computational thinking, where he explained that CT has been major component of computer science since the 1950s and so has the idea that CT can benefit people in a variety of fields. Unfortunately, Denning claims, recent attempts to make CT appealing to fields other than CS have led to "vague and confusing definitions of CT" (p. 33). Denning's two main criticisms of Wing's definition of CT include the absence of any mention of computational models as well as the suggestion that any sequence of steps constitutes an algorithm. Denning prefers, instead, to accept a definition of CT proposed by Alfred Aho, which he claims better embodies the notion of CT from computer science, computational science, as well as other fields such as the humanities, law and medicine.

In 2012, Aho defined CT quite succinctly as "the thought processes involved in formulating problems so their solutions can be represented as computational steps and algorithms" (p. 832). Aho explained that an important part of the CT thought processes involve finding the appropriate models of computation, and if there are none, then developing new ones. This view is exemplified in some of the mathematical modelling work by Michelle Wilkerson.

Wilkerson believes that computer science shares language with mathematics that can be used to represent models resulting in a description of patterns and processes that can make up scientific and engineered systems (Wilkerson & Fenwick, 2017). When describing CT, Wilkerson, and co-author Michele Fenwick, explain:

While mathematics focuses on quantities, computational thinking focuses on processes. Students engaged in the practice of computational thinking break a complex problem or process up into smaller steps in order to better understand, describe, or explain it (Wilkerson & Fenwick, 2017, p. 189).

Wilkerson works with having students use or build computational models and simulations in order to better understand scientific and engineered systems. This approach to CT would be considered by Shuchi Grover as a good example of integration CT in an effort to enable or enrich learning in other disciplines.

8. A TALE OF TWO (OR THREE OR FOUR OR FIVE) CTs

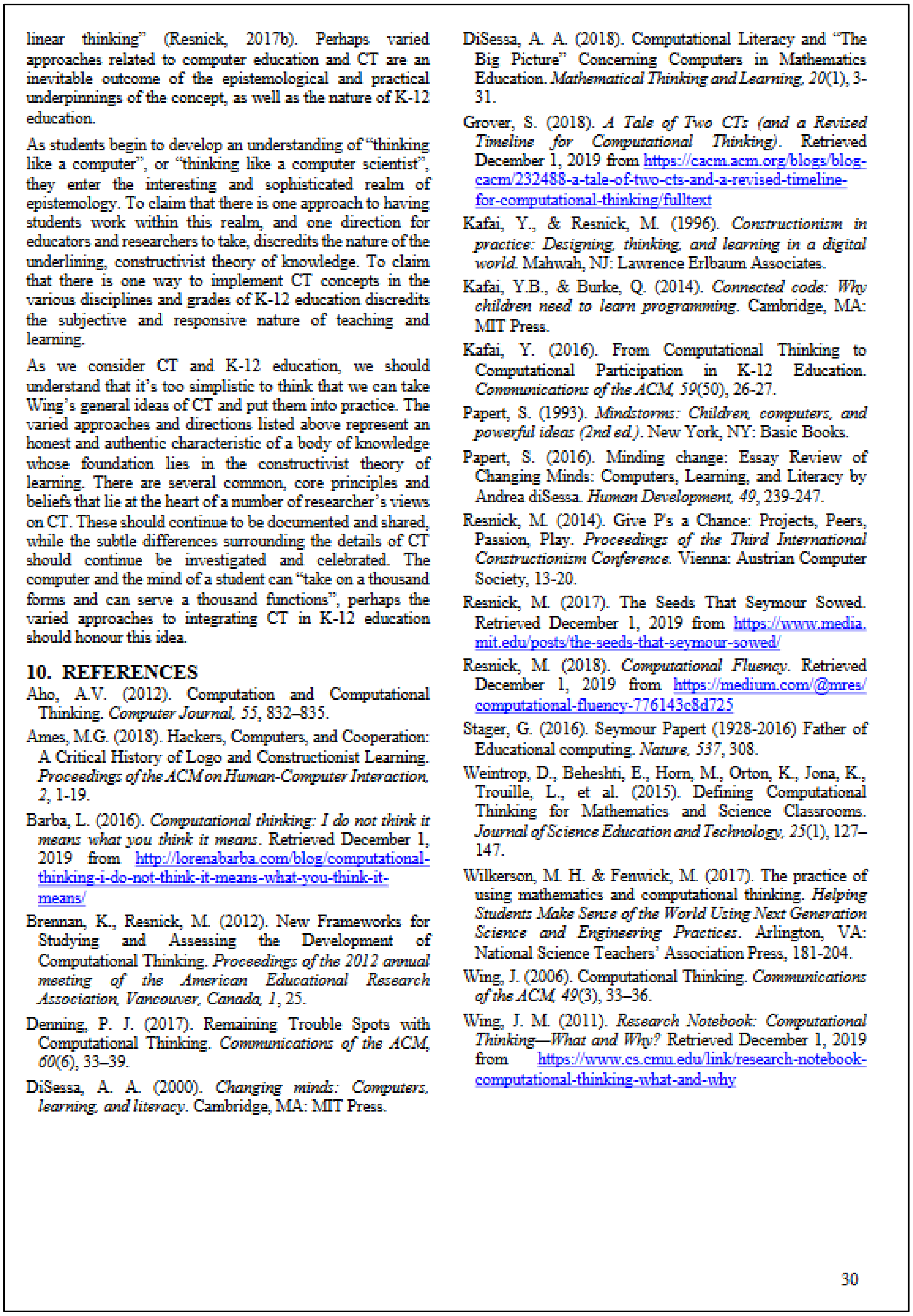
In A tale of two CTs (and a Revised Timeline for Computational Thinking) (2018), Grover argues that in order to make sense of CT in K-12 education we need to distinguish between main two views: computer science thinking in CS classrooms and CT in other disciplines. She explains that ideally, students will get a chance to experience CT in both settings during their K-12 schooling. Grover also presents a brief timeline of CT starting with the problem-solving practices discussed by G. E. Forsythe in 1968 and the elements of CS thinking discussed by Donald Knuth in the 1980s.

In regards to Wing, Grover credits her definition of CT for igniting K-12 computer science education and for calling attention to its role in other disciplines but also acknowledges that we should no longer be focused on "dreams of CT changing everyday behaviours of those who've learned this skill in curricular settings". Instead, we should view CT as playing a significant role in CS education and playing a role in helping students understand concepts within a variety of fields and disciplines.

9. CONCLUSION – MULTIPLE DIRECTIONS

While the varied approaches to CT may indicate disagreement on behalf of researchers in the field, it can also be a sign of the varied directions in which this powerful form of thinking can be taken. diSessa makes it clear that Computational Literacy is distinct from CT and that the field should have an analytical frame that can separate these ideas, and other CT ideas and movements (diSessa, 2018, p. 17). He goes on to explain that "it's not an issue of choosing terms; it is an issue of choosing directions" (diSessa, 2018, p. 17).

When deciding on how to frame an essay on Papert's ideas, Resnick acknowledged that it's "too simplistic to think that you can just take someone's ideas and put them into practice. Seymour was always skeptical about that type of top-down,



linear thinking" (Resnick, 2017b). Perhaps varied approaches related to computer education and CT are an inevitable outcome of the epistemological and practical underpinnings of the concept, as well as the nature of K-12 education.

As students begin to develop an understanding of "thinking like a computer", or "thinking like a computer scientist", they enter the interesting and sophisticated realm of epistemology. To claim that there is one approach to having students work within this realm, and one direction for educators and researchers to take, discredits the nature of the underlining, constructivist theory of knowledge. To claim that there is one way to implement CT concepts in the various disciplines and grades of K-12 education discredits the subjective and responsive nature of teaching and learning.

As we consider CT and K-12 education, we should understand that it's too simplistic to think that we can take Wing's general ideas of CT and put them into practice. The varied approaches and directions listed above represent an honest and authentic characteristic of a body of knowledge whose foundation lies in the constructivist theory of learning. There are several common, core principles and beliefs that lie at the heart of a number of researcher's views on CT. These should continue to be documented and shared, while the subtle differences surrounding the details of CT should continue be investigated and celebrated. The computer and the mind of a student can "take on a thousand forms and can serve a thousand functions", perhaps the varied approaches to integrating CT in K-12 education should honour this idea.

10. REFERENCES

Aho, A.V. (2012). Computation and Computational Thinking. *Computer Journal, 55*, 832–835.

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*, 1-19.

Barba, L. (2016). *Computational thinking: I do not think it means what you think it means*. Retrieved December 1, 2019 from http://lorenabarba.com/blog/computational-thinking-i-do-not-think-it-means-what-you-think-it-means/

Brennan, K., Resnick, M. (2012). New Frameworks for Studying and Assessing the Development of Computational Thinking. *Proceedings of the 2012 annual meeting of the American Educational Research Association, Vancouver, Canada, 1*, 25.

Denning, P. J. (2017). Remaining Trouble Spots with Computational Thinking. *Communications of the ACM, 60*(6), 33–39.

DiSessa, A. A. (2000). *Changing minds: Computers, learning, and literacy*. Cambridge, MA: MIT Press.

DiSessa, A. A. (2018). Computational Literacy and "The Big Picture" Concerning Computers in Mathematics Education. *Mathematical Thinking and Learning, 20*(1), 3-31.

Grover, S. (2018). *A Tale of Two CTs (and a Revised Timeline for Computational Thinking)*. Retrieved December 1, 2019 from https://cacm.acm.org/blogs/blog-cacm/232488-a-tale-of-two-cts-and-a-revised-timeline-for-computational-thinking/fulltext

Kafai, Y., & Resnick, M. (1996). *Constructionism in practice: Designing, thinking, and learning in a digital world*. Mahwah, NJ: Lawrence Erlbaum Associates.

Kafai, Y.B., & Burke, Q. (2014). *Connected code: Why children need to learn programming*. Cambridge, MA: MIT Press.

Kafai, Y. (2016). From Computational Thinking to Computational Participation in K-12 Education. *Communications of the ACM, 59*(50), 26-27.

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

Papert, S. (2016). Minding change: Essay Review of Changing Minds: Computers, Learning, and Literacy by Andrea diSessa. *Human Development, 49*, 239-247.

Resnick, M. (2014). Give P's a Chance: Projects, Peers, Passion, Play. *Proceedings of the Third International Constructionism Conference*. Vienna: Austrian Computer Society, 13-20.

Resnick, M. (2017). The Seeds That Seymour Sowed. Retrieved December 1, 2019 from https://www.media.mit.edu/posts/the-seeds-that-seymour-sowed/

Resnick, M. (2018). *Computational Fluency*. Retrieved December 1, 2019 from https://medium.com/@mres/computational-fluency-776143c8d725

Stager, G. (2016). Seymour Papert (1928-2016) Father of Educational computing. *Nature, 537*, 308.

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., et al. (2015). Defining Computational Thinking for Mathematics and Science Classrooms. *Journal of Science Education and Technology, 25*(1), 127–147.

Wilkerson, M. H. & Fenwick, M. (2017). The practice of using mathematics and computational thinking. *Helping Students Make Sense of the World Using Next Generation Science and Engineering Practices*. Arlington, VA: National Science Teachers' Association Press, 181-204.

Wing, J. (2006). Computational Thinking. *Communications of the ACM, 49*(3), 33–36.

Wing, J. M. (2011). *Research Notebook: Computational Thinking—What and Why?* Retrieved December 1, 2019 from https://www.cs.cmu.edu/link/research-notebook-computational-thinking-what-and-why

30

Appendix C. Initial codes from Thematic Analysis of preambles

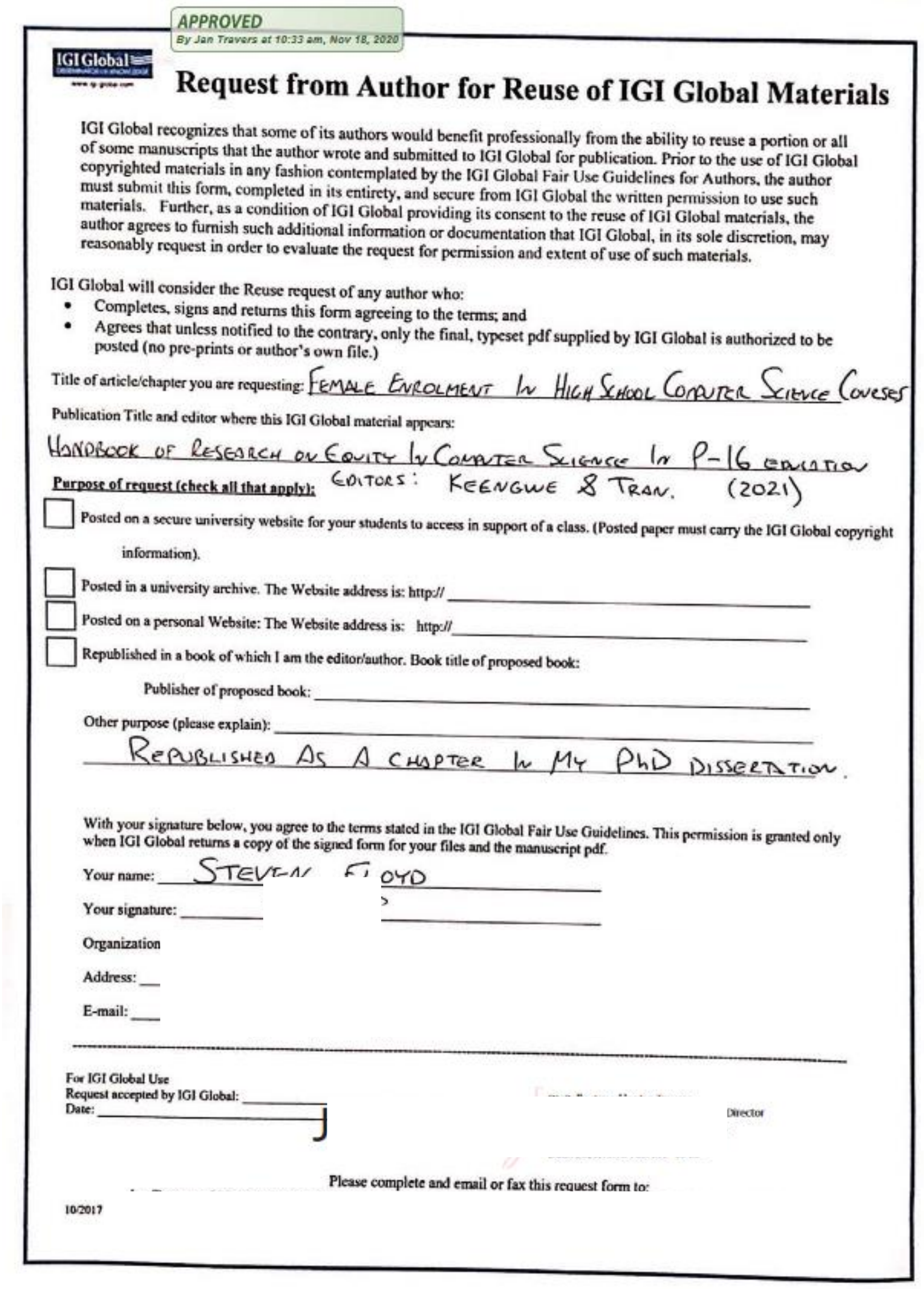
	Curriculum RP-33, Data Processing (1966)	Computer Science - Senior Division (1970)	Elements of Computer Technology (1970);	Informatics – Intermediate and Senior Division (1972);	Computer Studies - Intermediate and Senior Division (1983)	Computer Studies – Ontario Academic Course (1987);	The Ontario Curriculum Grade 11 and 12 - Technological Education (2000)	The Ontario Curriculum Grade 10 to 12 - Computer Studies (2008)
Impact of technology on society (22 references)	1	3	1	2	9	2	1	3
How curriculum document was designed/created (15 references)	0	1	4	7	2	0	1	0
Problem solving (12 references)	0	2	1	1	2	0	3	3
Training for future career (11 references)	1	0	5	1	0	0	1	3
How to teach the course(s) (9 references)	0	0	4	5	0	0	0	0
Student choice and differentiation in terms of depth (8 references)	0	3	1	2	2	0	0	0
Everyone needs a basic understanding of concepts (8 references)	0	1	3	0	2	0	1	1
Other courses and credits (7 references)	0	0	3	4	0	0	0	0
Post-secondary preparation (7 references)	0	0	0	2	0	1	0	4
New tools in society to store information (7 references)	1	0	0	2	3	0	1	0
Use of computer for creative pursuits (6 references)	0	0	0	1	3	0	1	1
Students need to be computer literate (6 references)	0	0	0	3	2	1	0	0
Only guidelines are provided (6 references)	1	1	0	4	0	0	0	0
Very technical aspects of Computer (5 references)	0	0	4	1	0	0	0	0
Computer as an object of study (5 references)	0	0	1	0	0	1	1	2
Cross-curricular and other subjects (5 references)	0	1	1	1	0	1	0	1

Students enjoy this work (5 references)	1	2	0	0	0	0	1	1
Other skills that can be developed (4 references)	0	0	1	0	0	0	1	3
How courses were developed (4 references)	1	0	0	2	1	0	0	0
Development of transferable skills (4 references)	0	0	0	0	0	0	2	2
Computer programming and large program design concepts (4 references)	0	0	0	0	0	0	1	3
How the computer represents objects (4 references)	0	0	0	0	0	0	3	1
Ethics and appropriate use of technology (4 references)	0	0	0	0	3	0	0	1
Computational Thinking and explaining the problem to the computer (4 references)	0	1	0	0	1	1	1	0
Changed how we think about problems and how to solve them (4 references)	0	2	1	0	0	1	0	0
Computers can extend human capabilities (4 references)	0	2	0	0	1	1	0	0
Consider local/student needs before implementing course (4 references)	0	2	1	0	0	1	0	0
Need for experimentation to establish best pedagogical practice (4 references)	3	1	0	0	0	0	0	0

Appendix D. Themes developed through Thematic Analysis of preambles.

	Curriculum RP-33, Data Processing (1966)	Computer Science - Senior Division (1970)	Elements of Computer Technology (1970);	Informatics – Intermediate and Senior Division (1972);	Computer Studies - Intermediate and Senior Division (1983)	Computer Studies – Ontario Academic Course (1987);	The Ontario Curriculum Grade 11 and 12 - Technological Education (2000)	The Ontario Curriculum Grade 10 to 12 - Computer Studies (2008)
Use computers to automate tasks or solve problems (31 references)	0	9	4	2	3	2	4	7
Post-secondary, training and careers (21 references)	3	1	3	4	2	1	1	6
Impact of technology on society (19 references)	4	1	1	1	7	2	1	2
Dynamic nature of technology in society and education (16 references)	1	2	0	7	4	1	0	1
How the course is structured (15 references)	0	2	2	4	4	0	1	2
Computer as an object of study (13 references)	0	0	3	2	0	0	4	5
Differentiate for needs of students (13 references)	1	4	2	2	4	0	0	0
Teacher flexibility & experimentation (9 references)	2	1	1	5	0	0	0	0
Content simply a guide for educators (8 references)	1	2	2	3	0	0	0	0
learn basic understanding of computers (8 references)	0	2	3	2	0	0	0	0
Ethics (8 references)	0	0	0	2	4	0	0	2
Student enjoyment and interest (7 references)	1	2	0	1	0	0	2	1
Creativity (7 references)	0	0	0	1	4	0	1	1
How the course was developed (6 references)	1	0	4	0	1	0	0	0
Cross curricular connections (6 references)	0	1	2	1	0	1	0	1
General computer literacy for all (6 references)	0	0	0	2	3	0	1	1
Developing programs (6 references)	0	0	0	0	0	0	1	5

Appendix E. Letter of permission to reprint contents in Chapter 4.



APPROVED
By Jan Travers at 10:33 am, Nov 18, 2020

IGI Global

Request from Author for Reuse of IGI Global Materials

IGI Global recognizes that some of its authors would benefit professionally from the ability to reuse a portion or all of some manuscripts that the author wrote and submitted to IGI Global for publication. Prior to the use of IGI Global copyrighted materials in any fashion contemplated by the IGI Global Fair Use Guidelines for Authors, the author must submit this form, completed in its entirety, and secure from IGI Global the written permission to use such materials. Further, as a condition of IGI Global providing its consent to the reuse of IGI Global materials, the author agrees to furnish such additional information or documentation that IGI Global, in its sole discretion, may reasonably request in order to evaluate the request for permission and extent of use of such materials.

IGI Global will consider the Reuse request of any author who:

- Completes, signs and returns this form agreeing to the terms; and
- Agrees that unless notified to the contrary, only the final, typeset pdf supplied by IGI Global is authorized to be posted (no pre-prints or author's own file.)

Title of article/chapter you are requesting: FEMALE ENROLMENT IN HIGH SCHOOL COMPUTER SCIENCE COURSES

Publication Title and editor where this IGI Global material appears:

HANDBOOK OF RESEARCH ON EQUITY IN COMPUTER SCIENCE IN P-16 EDUCATION

Purpose of request (check all that apply): EDITORS: KEENGWE & TRAN. (2021)

☐ Posted on a secure university website for your students to access in support of a class. (Posted paper must carry the IGI Global copyright information).

☐ Posted in a university archive. The Website address is: http:// ________

☐ Posted on a personal Website: The Website address is: http://________

☐ Republished in a book of which I am the editor/author. Book title of proposed book:

Publisher of proposed book: ________

Other purpose (please explain): ________

REPUBLISHED AS A CHAPTER IN MY PhD DISSERTATION.

With your signature below, you agree to the terms stated in the IGI Global Fair Use Guidelines. This permission is granted only when IGI Global returns a copy of the signed form for your files and the manuscript pdf.

Your name: STEVEN FLOYD

Your signature: ________

Organization

Address: ___

E-mail: ___

For IGI Global Use
Request accepted by IGI Global: ________
Date: ________

Director

Please complete and email or fax this request form to:

10/2017

Curriculum Vitae

Name:	Steven Paul Floyd
Post-secondary Education and Degrees:	Western University London, Ontario, Canada 1997-2001 Bachelor of Arts (Honours)
	The University of Western Ontario London, Ontario, Canada 2002-2003 Bachelor of Education
	The University of Western Ontario London, Ontario, Canada 2004-2009 Master of Education
	The University of Western Ontario London, Ontario, Canada 2018-2022 Doctor of Philosophy
Honours and Awards:	Dean's Honor List, Graduate with Distinction 1998-2003
	Infosys Foundation USA, the Association for Computing Machinery and the Computer Science Teachers Association Award for Teaching Excellence in Computer Science 2017
	ICER '19 International Computing Education Research Conference Doctoral Consortium Grants 2019
	Centre for Inclusive Education Research Award 2019
	Scholarship Programme - International Conference on Computational Thinking Education 2020 2020
	Province of Ontario Graduate Scholarship 2021-2022

Related Work Experience	Education Officer – Ontario Ministry of Education 2020-Present
	Instructor – Computational Modelling in Mathematics and Science Education (B.Ed. course 5467) 2021-2022, 2019-2020
	e-learning course writer, reviewer, editor, technological pedagogy expert – Ontario Ministry of Education 2008-2018
	Secondary Teacher – London District Catholic School Board 2003-2018

Publications:

Gadanidis, G., Scucuglia, R. Hughes, J., Namukasa, I., & Floyd, S. (In Press). Computational literacy and mathematics education [Special Issue]. *ZDM, ?(3)*, ??-??.

Floyd. S. (2021). Female enrollment in high school computer studies courses. In J. Keengwe & Y. Tran (Eds.), *Handbook of research on equity in computer science in P-16 education (pp. 31-43)*. IGI Global.

Floyd, S. (2020). Computational thinkers: Contemporary Approaches and directions in computational thinking for K-12 education. In *Proceedings of International Conference on Computational Thinking Education 2020 (pp. 27-32)*. Hong Kong: The Education University of Hong Kong.

Grover, S., & Floyd, S. (2020). Questions and inquiry. In S. Grover (Ed.), *Computer science in K-12: An a to z guide on teaching programming* (pp. 180-188). Edfinity.

Floyd, S. (2019). Doors, walls and windows? The gender gap in Ontario high school computer science. In *Proceedings of the 2019 ACM Conference on International Computing Education Research* (pp. 301-301). ACM.

Floyd, S. (2019). A qualitative content analysis of K-8 coding curriculum. In *Proceedings of the 2019 ACM Conference on International Computing Education Research* (pp. 329-330). ACM.

Floyd, S. P. (2019, February). Historical high school computer science curriculum and current K-12 initiatives. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education* (pp. 1287-1287). ACM.

Floyd, S. P., & Sorbara, L. (2019, February). Sports analytics as a context for computational thinking in K-12 education. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education* (pp. 1282-1282). ACM.