Integrate coding into Ontario elementary mathematics teaching and learning:

A curriculum analysis

© *Yimei Zhang*

Department of Integrated Studies in Education, McGill University

A thesis submitted to McGill University in partial fulfillment of the requirements of the degree of Master of Arts in Education and Society

Contents

Tables

Figures

Abstract

Background: In a digital society, coding has emerged as a critical skill, essential for equipping citizens with the competencies needed for the 21st century (Tuomi et al., 2018). Recognizing its value, scholars have promoted the integration of coding into education to foster students' 21st century skills (Gretter & Yadav, 2016). In Canada, coding has also been regarded as an important skill (Francis et al., 2017). Much work has been done to promote coding integration in K–12 education, especially in Ontario, where coding has been integrated into the elementary mathematics education curriculum (Ontario, 2022; Ministry of Education of Ontario, 2020). Therefore, this research aims to investigate (1) how Ontario ministry of education conceptualized coding within their curricula, and (2) how Ontario Math Support integrated coding into their lesson plans.

Conceptual Framework: In this study, I created a conceptual framework, named Coding Content Knowledge for Mathematics Teaching and Learning (CCKMTL) model. This model combined curriculum analysis (Posner, 2004), mathematics content knowledge (Ball et al., 2008), and a pedagogical framework for integrating computational thinking (Kotsopoulos et al., 2017), offering a multifaceted lens through which to investigate coding integration into the Ontario elementary mathematics curricula.

Methodology: This research employed a qualitative study methodology. Data were collected from Ontario curriculum documents and Ontario Math Support's lesson plans.

Results: The findings revealed that Ontario's educational framework systematically incorporated coding at various educational stages, which enabled students to apply coding to mathematical challenges as well as other transdisciplinary problems (e.g., music, finance).

Implications: The study has practical and theoretical implications. Practically, it provides valuable reference for multiple communities, including in-service, pre-service teachers, school administrators, policymakers, and educational researchers on how to integrate coding into mathematics teaching and learning. Theoretically, this study has expanded the computational thinking model developed by Kotsopoulos et al. (2017) by identifying related themes and subthemes. It potentially broadens its scope and applicability within the context of elementary mathematics. Therefore, future research plans to use the CCKMTL framework developed in this study to investigate the design of curriculum policies that have effectively integrated coding into various educational programs, such as mathematics, science, music, etc.

Résumé

Contexte: Dans une société numérique, le codage est apparu comme une compétence essentielle, indispensable pour doter les citoyens.nes des compétences nécessaires pour le 21e siècle (Tuomi et al., 2018). Reconnaissant sa valeur, des universitaires promeuvent l'intégration du codage dans l'éducation pour favoriser les compétences du 21e siècle des étudiants universitaires (Gretter & Yadav, 2016). Au Canada, le codage a également été considérée comme l'une des compétences les plus importantes (Francis et al., 2017). De nombreux travaux ont été réalisés pour promouvoir l'intégration du codage dans l'éducation K-12, en particulier en Ontario, où le codage a été intégré dans l'enseignement des mathématiques du primaire (Ministère de l'Éducation de l'Ontario, 2020). Ainsi, cette recherche vise à étudier (1) comment le ministère de l'Éducation de l'Ontario a conceptualisé le codage dans leurs programmes d'études, et (2) comment l'équipe de soutien en mathématiques de l'Ontario a intégré le codage dans leurs plans de leçon.

Cadre Conceptuel: Cette étude a créé un cadre conceptuel, nommé modèle de Connaissance du Contenu du Codage pour l'Enseignement et l'Apprentissage des Mathématiques (CCCKMTL). Ce modèle combine l'analyse du curriculum (Posner, 2004), la connaissance du contenu mathématique (Ball et al., 2008), et un cadre pédagogique pour intégrer la pensée computationnelle (Kotsopoulos et al., 2017), offrant une lentille multifacette à travers laquelle étudier l'intégration du codage dans l'enseignement des mathématiques du primaire en Ontario.

Méthodologie: Cette recherche a employé une méthode d'étude qualitative. Les données ont été collectées à partir de documents du curriculum de l'Ontario et des ressources de soutien en mathématiques de l'Ontario.

Résultats: Les résultats ont révélé que le cadre éducatif de l'Ontario intégrait systématiquement le codage à divers stades éducatifs, ce qui permettait aux élèves d'appliquer le codage à des défis mathématiques ainsi qu'à des problèmes transdisciplinaires (par exemple, musique, finance).

Implications: L'étude a des implications pratiques et théoriques. D'un point de vue pratique, elle fournit une référence précieuse pour plusieurs communautés, y compris les enseignants en service et en formation, les administrateurs scolaires, les décideurs politiques et les chercheurs en éducation sur comment intégrer la programmation dans l'enseignement et l'apprentissage des mathématiques. Théoriquement, cette étude a élargi le modèle de pensée computationnelle développé par Kotsopoulos et al. (2017) en identifiant des thèmes et sous-thèmes liés. Elle élargit potentiellement sa portée et son applicabilité dans le contexte des mathématiques élémentaires.

Abbreviations

- AR: Academic Resilience
- CA: Curriculum Analysis
- CCKMTL: Coding Content Knowledge for Mathematics Teaching and Learning
- CT: Computational Thinking
- PCK: Pedagogical Content Knowledge
- SMK: Subject Matter Knowledge

Contributions of Authors

I am the sole author of this thesis under the guidance and support of my thesis supervisor, Dr. Annie Savard, throughout the entire process (e.g., writing and revision). Specifically, during the revision process, I carefully addressed all the comments and suggestions from my reviewer, Dr. Hannah Chestnutt. Additionally, I received valuable guidance and support from Dr. Marta Kobiela and Dr. Allison Gonsalves. Below, I detail the contributions and roles of everyone involved.

Before the study

To start this master's thesis, I created a thesis outline and obtained approval from my thesis supervisor Dr. Annie Savard.

Chapter 1

The first version of Chapter 1 was completed by myself with the support of my supervisor Dr. Annie Savard. Specifically, under the guidance of Dr. Annie Savard, I formulated two research questions that guided this study. After reviewing literature on coding and mathematics education, I justified situating this study within the Ontario curricula. Additionally, during the revision process, I carefully revised Chapter 1 based on my reviewer's (Dr. Hannah Chestnutt) comments with constant guidance and support from Dr. Annie Savard and Dr. Marta Kobiela.

Chapter 2

In Chapter 2, I found the useful literatures under the guidance of my supervisor Dr. Annie Savard. Furthermore, during the revision process, I carefully revised Chapter 2 based on my

reviewer's (Dr. Hannah Chestnutt) comments with the guidance and support from Dr. Annie Savard and Dr. Marta Kobiela.

Chapter 3

In Chapter 3, I constructed my conceptual framework with the assistance of my supervisor, Dr. Annie Savard. Furthermore, during the revision process, I carefully addressed the suggestions brought up by my reviewer, Dr. Hannah Chestnutt under the guidance of Dr. Annie Savard and Dr. Marta Kobiela.

Chapter 4

This chapter focused on methodologies, data collection, and analysis. I carefully revised this chapter based on Dr. Hannah Chestnutt's suggestions and received guidance from Dr. Annie Savard and Dr. Marta Kobiela.

Chapters 5 and 6

In Chapter 5, I reported the results from Ontario ministry of education and Ontario Math Support, respectively. Before revision, my supervisor Dr. Annie Savard provided comments and feedback on the areas needed to be improved and elaboration. During the revision process, I revised these parts based on my reviewer, Dr. Hannah Chestnutt's, comments. Additionally, I received constant feedback and support from Dr. Annie Savard and Dr. Marta Kobiela.

Chapter 7

I discussed the two research questions and pointed out the weakness and strengths of my conceptual framework. Additionally, I discuss the limitations of my research. I wrote this Chapter and obtained feedback from my supervisor. During the revision process, I carefully

addressed the issues brought up by the reviewer and received feedback and guidance from Dr. Annie Savard and Dr. Marta Kobiela.

Chapter 8

In the final Chapter, I summarized the research findings, contributed to existing research, discussed implications for various communities, and outlined future research directions.

Acknowledgement

Firstly, I would like to express my deepest gratitude to my supervisor Dr. Annie Savard, for her invaluable guidance, patience, and support through the journey of my thesis. Her insights and feedback have been instrumental in shaping this work. I am also grateful for the time she managed to squeeze out of her busy schedule to meet me, help polish and revise my work, and provide continuous encouragement and direction.

I am also thankful to my reviewer, Dr. Hannah Chestnutt, whose rigorous evaluation, and constructive comments have significantly improved my thesis. Her expertise and attention to detail have helped evaluate the quality of this work.

Furthermore, I want to extend my thanks to Dr. Marta Kobiela, and Dr. Allison Gonsalves, for their academic and emotional support through the revision of the thesis. Their encouragement and wise counsel during the moments of struggling and challenges have been a source of inspiration. Their persistent support in guiding me through difficult times have been indispensable to my journey.

Finally, I want to reserve my gratitude for my family: my mom, Haixia Wang, my dad, Zhen Zhang, my boyfriend, Dr. Luhua Xu, and my best friend, Xiye Li. Their belief in me has supported me all the time. Without their unconditional love and encouragement, this journey would have been much more challenging. Their constant support and understanding have been the core of my success.

Chapter 1: Introduction

1.1 Research Background

Coding, the act of writing code that tells a computer or machine what to do, is an essential part of computer science (Van Roy & Haridi, 2004). However, coding has now transcended its traditional boundaries of computer science and is now recognized as a valuable skill for various disciplines (Demirer & Sak, 2016). Its applications range from data analysis and automation to web development and scientific research, and scholars in social sciences use coding for text mining, sentimental analysis, data visualization, and digital archiving (Mäntylä et al., 2018). Because of this versatility, coding is recognized as one of the core competencies of "21st-century skills" (Kanbul & Uzunboylu, 2017).

Recognizing its importance, several countries have taken steps to integrate coding into their educational systems (Fraillon et al., 2020). In Singapore, for instance, the Ministry of Education has developed a curriculum to enhance computational thinking (CT) and programming abilities across all levels, from primary to tertiary (Seow et al., 2019). Similarly, in 2022, the Ministry of Education of the People's Republic of China announced that they would launch an assessment on K-12 students' information literacy and implement artificial intelligence-related courses in K-12 education, thus integrating coding and programming into education (Ministry of Education of the People's Republic of China, 2022). Moreover, most primary schools in China have launched a compulsory course for students called "Basic Information Technologies," which introduces basic coding concepts (Jiang & Li, 2021). More recently, in 2024, coding will become part of the national computing curricula (from ages 5 to 14) in England (Jones et al., 2021).

In Canada, some provinces have significantly invested in coding education, recognizing its value in fostering problem-solving, teamwork, critical thinking, and innovation. For example, British Columbia introduced coding into the K–12 curriculum in 2016, with the government funding professional development programs to support teachers in integrating coding into their lesson plans (Burgmann, 2016). In addition, Nova Scotia's Action Plan for Education (2017) highlights coding as a key competency and introduces mandatory coding courses in middle school technology education (Julie, 2017).

These initiatives across countries emphasize a global movement towards integrating coding in education to prepare students for a digital future. This raised my interest in studying how to integrate coding into educational contexts. To explore further, I decided to investigate coding integration in Canadian educational contexts.

1.2 Why Focus on Coding Integration?

Before delving into the rationale of investigating coding integration in Canadian educational contexts, it is essential to define what coding integration entails in this study. According to Dralle-Moreano's statement (2021), coding integration refers to systematic incorporation of computer programming principles and activities into educational contexts. It encompasses the use of coding languages and tools to enhance teaching and learning experiences (Dralle-Moreano, 2021).

Prior research has shown coding integration can support students' learning process. For instance, Tugun et al. (2017) conducted experimental research to examine how to integrate coding into 9th grade classrooms. They found that flipped classrooms were ideal environments for students to learn coding. Additionally, Egbert et al. (2021) investigated how early elementary

school students could learn coding in the classrooms. Their findings demonstrated that coding activities (i.e., robots and well-designed age-appropriate coding tasks) could motivate students and engage students in the learning process (Egbert et al., 2021). Much of these studies focused on classroom settings. However, there is limited research investigating coding integration within the curricula. Therefore, this research focuses on how to integrate coding into curricula. Specifically, I focus on coding integration in Canadian curricula, concentrating on coding integration into Ontario elementary mathematics curricula.

Canadian provinces and territories are responsible for their own educational systems and programs, and thus coding integration varies across Canada. My main rationale for focusing on Ontario is that this province is the most populous province in Canada, constituting 38.3 % of the country's population (Ontario, 2023). Most importantly, the Ministry of Education of Ontario has integrated coding into the elementary mathematics curricula (i.e., Grades 1 to 8) from 2020. This integration signifies a major step preparing the future generation for the digital society from an early age.

Specifically, there are several reasons why incorporating coding into education at early age is crucial. First, integrating into elementary mathematics can lead to more engagement for young learners than traditional learning methods because coding projects are interactive (Govind et al., 2020). For example, Videnovik et al. (2020) used two specific tools: the "Scotti Go!" board game and the BBC "Micro:bit" platform to teach coding in primary schools. Their findings indicated that integrating coding into primary education could improve students' critical and problem-solving skills and raised their interest in learning and motivation (Videnovik et al., 2020). Second, early exposure to coding can prepare students for a future in which digital literacy is a prerequisite for many jobs (Welch et al., 2019), such as machine learning engineers,

data scientists, and software developers (Khuraisah et al., 2020). Third, coding tools at the elementary level can encourage teamwork and help students develop social skills, such as communication and collaboration, as they work together to solve problems (Gim, 2021). For instance, Calder and Rhodes (2021) investigated how ScratchMaths (a collaborative problem-solving platform) facilitated students' learning processes and their findings indicated that coding could improve primary students' collaborative learning in elementary mathematics.

Overall, in Ontario, the strategic inclusion of coding within the elementary mathematics curricula not only aligns with the evolving demands of the 21st century (Lafee, 2017) but also fosters a more interactive and collaborative learning environment for young learners (Calder & Rhodes, 2021). Considering these advantages, my master's research explored how coding was integrated in the Ontario elementary mathematics curricula.

1.3 Research Objectives and Questions

In this thesis, I analyzed the ways that coding was integrated into Ontario education curricula. There are two research questions guiding this study:

(1) How was coding conceptualized in the Ontario elementary mathematics curricula?

(2) How was coding integrated into Ontario Math Support's lesson plans?

Specifically, to answer the first research question, I delved into the conceptual foundation of coding in the Ontario curricula, aiming to understand the educational values, skills, and competencies that coding was intended to develop in students. This involved analyzing Ontario curriculum documents to identify its goals, definitions, and expectations set forth by educational policymakers regarding coding education within Ontario mathematics contexts. It was a critical

and first step in appreciating the broad context of why coding should be integrated into the mathematics curricula. Additionally, to address this question, I applied case study approach (i.e., Case 1, Grades 1 to 3; Case 2, Grades 4 – 6; Case 3, Grades 7 – 8) to capture the diversity of coding concepts and activities across different educational levels. The result of this question is presented in Chapter 5, and the discussion for it is displayed in Chapter 7.

The second research question transitioned from the theory to practice by examining how these conceptual goals of coding integration were operationalized by Ontario Math Support's lesson plans. There are two main reasons why I chose the resources provided by Ontario Math Support. Firstly, the resources are closely aligned with the Ontario Ministry of Education's curriculum guidelines. Secondly, Ontario Math Support is known for its high-quality resources that are freely accessible to teachers (Ontario Math Support, 2024). This accessibility ensures that all students, regardless of their socio-economic backgrounds, can benefit from these well-developed educational materials. To answer this question, I adopted a CT framework developed by Kotopoulos et al. (2017) and conducted content analysis to examine how coding was integrated into the mathematics lesson plans. The result of this question is presented in Chapter 6, and the discussion for it is shown in Chapter 7. In the next section, I will present an overview of the thesis.

1.4 Overview of the Thesis

This master's thesis consists of eight chapters. In the introduction, which is the first Chapter, I presented the research background and the research questions of the study. In Chapter 2, I present a literature review to illustrate the relationships between coding, teaching, and learning mathematics. In Chapter 3, I outline a conceptual framework for conducting this

research. Chapter 4 covers the methodology employed in my study. The results of my research are presented in Chapter 5 and 6, each focusing on a specific dataset – Ontario official curricula (Chapter 5) and lesson plans provided by Ontario Math Support (Chapter 6). In Chapter 7, I discuss the two research questions, and point out the weakness and strengths of the conceptual framework and the limitation of the research. In Chapter 8, I revisit the findings while explicitly detailing the contributions of this study. Specifically, this study enriches the existing literature on curriculum studies and CT. Finally, I present the implications and future research directions.

Chapter 2: Literature Review

Firstly, in this chapter I provide a more detailed description of how I am conceptualizing coding in this thesis by describing the relationship between coding and computational thinking (CT). Then, I examine the intersection of coding and mathematics education in the research literature to show how coding and mathematics have been integrated in prior research.

2.1 Defining Coding and Computational Thinking

Coding has broadly been conceptualized in two ways. The first way has typically focused on *coding skills* involved in writing a programming language rather than the theoretical concepts, perspectives, and contents that emerge from mathematics education and other disciplines (Albion, 2016). For example, a skill-based definition of coding may focus on coding as the acquisition of programming languages (e.g., Java, Python, C++, and HTML) (Albion, 2016). In contrast, others have suggested an *expansive view* of coding that goes beyond programming and can be seen as a combination of skills, such as problem solving and design thinking (Tuomi et al., 2018). For instance, when coding, a UX (i.e., user experience) designer needs not only to use programming languages (HTML) but also to understand the user needs, conceptualize solutions, and iteratively refine their products (e.g., Apps, Games) using critical and analytical thinking (User experience designer, 2024). It demonstrates that coding is multifaceted, blending technical skills with complex cognitive processes aimed at solving problems and expressing intricate ideas.

This expansive view of coding aligns with the concept of CT, which also transcends mere programming (Shute et al., 2017). CT is defined as the application of computer science concepts

to solve problems and develop technological fluency (Barr et al., 2011; Shute et al., 2017; Wing, 2006). Shute et al. (2017) described CT as a problem-solving process that entails the use of core computer science principles, such as decomposition, abstraction, algorithms, debugging, iteration, and generalization (Shute et al., 2017). These principles are not only fundamental to coding but also crucial for engaging with and solving a wide array of problems.

Moreover, CT extends beyond computer science; it shares similarities with mathematical thinking (e.g., problem solving) (Rich et al., 2019), engineering thinking (designing and evaluating processes) (Grover, 2011), and scientific thinking (systematic analysis) (Weintrop et al., 2016). This interdisciplinary nature emphasizes CT's role as a bridge, linking computer science with broader educational and cognitive domains. Given the similarities between the definitions of coding and CT presented above, this study regards CT as a specific form of coding.

2.2 Integrating Coding and Mathematics

My goal for this section is to synthesize the current debates and research about the relationships between coding and mathematics education. After reviewing the current literature, two themes about the ways in which coding and mathematics have been integrated in lessons: (1) Coding as contexts for improving mathematics competencies and (2) Mathematics as contexts for improving coding competencies. Considering that the words "contexts" and "competencies" are broad terms, I will define the terms **"contexts"** and **"competencies"**, and present how I will use these two terms in Themes 1 and 2.

The term, *contexts*, refers to the circumstances or settings in which an event or activity occurs (Mishler, 1979). It includes the conditions that influence the nature of learning or teaching (Edwards et al., 2009). In educational terms, a context can refer to the specific setting in which

learning takes place, including the classroom environment, workplace, etc. (Thornton Moore, 2004). In this study, the coding contexts (Theme 1) refer to the scenarios in which coding is taught and learned, ranging from conducting coding tasks in the classrooms to solving real-life problems (e.g., developing website pages, applying machine learning to predict students' performance) (Videnovik, et al., 2021). The mathematic contexts (Theme 2) refer to settings in which mathematics concepts are taught and learned, ranging from traditional classroom lessons to hands-on problem-solving sessions (Boaler, 1993).

The term, *competencies*, refers to the knowledge, skills, attitudes, and behaviors that students are expected to develop in a subject area (Albanese et al., 2008). Skills include hard skills, such as the ability to perform specific tasks (e.g., solving mathematics problems, or designing a computer program), and soft skills, including collaboration and creativity (Vogler et al., 2018). In this study, the mathematics competencies (Theme 1) include two components: (1) *specific mathematics skills*, which are the cognitive aspects of learning mathematics (e.g., reasoning and, problem solving) (Karagiannakis et al., 2014) and which can be regarded as hard skills, and (2) *mathematics attitudes*, which refer to the affective domain of learning mathematics, encompassing learners' emotions, beliefs, and attitudes towards mathematics (e.g., self-efficacy, enjoyment, and math anxiety) (Vukovic et al., 2013; Hoffmann, 2010) and which can be regarded as soft skills. Coding competencies (Theme 2) also comprise two components: (1) *coding skills*, which refer to the technical abilities required to write, test, and debug software or algorithms (e.g., debugging and pattern recognition) and which can be regarded as hard skills (Kalyenci et al., 2022), and (2) *coding attitudes*, which encompass the affective dimensions related to coding (e.g., confidence, persistence, and creativity) (Mason & Rich, 2020) and which can be regarded as soft skills.

By defining contexts and competencies, I gain a structured lens through which to examine the relationships between coding and mathematics. Next, I will present the studies related to Theme 1.

2.2.1 Theme 1: Coding as Contexts for Improving Mathematics Competencies

The first theme that I identified in the research literature was that coding contexts could enhance students' mathematics competencies, more specifically the skills and attitudes. That is, in these studies, researchers or educators involved students in coding tasks and then found benefits for mathematical competency development. I begin with a review of the existing literature on the influence of coding contexts on mathematics education.

To begin, within coding contexts, various mathematics skills can be developed. One set of fundamental mathematical skills that has been shown to develop in coding contexts are measurement skills (Ceylan & Aslan, 2023). Measurement skills involve understanding and applying units of measurement, as well as comparing and estimating lengths, weights, and volumes (Welch et al., 2022; Browning et al., 2014). For example, a study conducted by Ceylan and Aslan (2023) employed a quasi-experimental research design to examine how to improve preschool students' mathematics measurement skills within coding programs. Specifically, in the coding programs, the student used robotics to understand what length measurement, area, and volume were. They found that students who participated in these coding programs (i.e., robotics) exhibited significant improvement in measurement skills (Ceylan & Aslan, 2023). In addition to measurement skills, patterning skills can also be developed within coding contexts (Miller, 2019). For example, Miller (2019) applied coding (i.e., six lessons, three with a focus of Scratch and three with a focus of coding robots) to improve Grade 2 students' mathematics pattern

recognition skills and found that the students gained a better understanding and recognition of mathematical patterns and structures in coding contexts compared to students in traditional contexts (Miller, 2019). A third set of mathematical skills that coding can develop are geometric skills. For instance, a study conducted by Iskrenovic-Momcilovic (2020) highlighted the effectiveness of Scratch (a high-level, block-based programming language) in teaching and learning geometry. Specifically, in the study, the students were required to create lines and complex geometric shapes using block-like interface. The study found that engaging in coding using the Scratch program greatly improved students' understanding of geometry concepts (e.g., the concepts of a quadrilateral, triangles, angles) (Iskrenovic-Momcilovic, 2020). These findings collectively emphasized the potential of coding contexts in enhancing students' mathematics skills. By facilitating deeper engagement with complex mathematical concepts through coding contexts, students can improve these specific mathematical skills.

Beyond mathematics skills enhancement, integrating coding into mathematics can positively influence students' attitudes toward the subject. For example, a study conducted by Heinzman (2022) found that high school students felt a great sense of a community after coding training and were more confident and empowered to learn mathematics. Furthermore, this study demonstrated that coding could significantly enhance high school students' engagement, self-efficacy, and interest in learning mathematics (Heinzman, 2020). Heinzman's findings were echoed by Ke's (2014) study, which explored the effectiveness of designing educational computer games on students' dispositions toward mathematics. Ke (2014) observed that coding activities helped students develop more positive attitudes towards mathematics, marked by increased self-confidence, value, enjoyment, and motivation. Together, these studies highlighted the powerful role of coding in redefining how students perceived and engaged with mathematics,

presenting a promising direction for educational strategies aimed at enhancing mathematics attitudes within coding contexts.

Whereas the above studies have treated coding contexts as programming environments, recently, the exploration of the impact of coding contexts on students' mathematics competencies has expanded to include a specific focus on CT (Grover & Pea, 2018; Grover & Pea, 2013). For example, Gadanidis et al. (2017) designed a course that aimed to integrate CT into elementary mathematics teacher education. This course was developed to help students (pre-service teachers) acquire a deeper conceptual understanding of mathematics and CT-based teaching methodologies. Their findings showed that the inclusion of CT ignited students' (i.e., pre-service teachers') interest and creativity but also motivated them to embed coding contexts into their future mathematics instruction.

Despite these positive outcomes, some studies have cautioned against overestimating the efficacy of coding contexts in improving mathematics competencies. For example, Kalelioğlu (2015) reported the limited effect of coding contexts on improving primary students' mathematical reflective thinking. Specifically, in this study, the researcher designed a 5-week course based on Code.org (i.e., a blocky coding platform) for 4th grade students to improve their mathematics reflective thinking skills toward problem solving. However, there was no significant difference between the pre-test and post-test results of students' performance after the coding course. This outcome suggested that the programming course based on Code.org might not significantly enhance primary students' mathematics reflective thinking skills. Similarly, Bernardo and Morris (1994) found no effect of coding contexts on improving high school students' mathematics modeling and procedural comprehension. They created a BASIC programming course intended to teach students programming languages. Yet, no significant

differences emerged between the treatment group (i.e., the BASIC programming course) and the control group (i.e., a traditional class) in terms of mathematics modeling or procedural comprehension test scores. These results underscore the complexity of integrating coding into mathematics education and highlighted the need for further research to better understand the different ways in which coding is integrated.

In summary, despite these studies noting limited efficacy of coding contexts on mathematical competencies, there is ample evidence that coding contexts have potential to significantly improve students' mathematical competencies. In theme 2, my focus will shift to a different way that coding has been integrated with mathematics. I will present research that shows how mathematics can serve as contexts for improving coding competencies.

2.2.2 Theme 2: Mathematics as Contexts for Improving Coding Competencies

In this section, I examine a second way that coding and mathematics have been integrated in prior research: by using mathematics contexts to improve students' coding competencies. I present research that has shown the use of mathematics contexts for developing coding skills (including CT skills) and coding attitudes.

Whereas in the research in Theme 1, students learned mathematics by engaging in *coding* tasks, this second body of work shows how coding competencies can be learned by engaging in *mathematics* tasks. Researchers have suggested benefits to using mathematics contexts (tasks) instead of coding contexts (tasks). For example, previous research has shown that coding education in schools has been limited, because of the constrained budgets (Boyabatlı et al., 2016), insufficient training for teachers (Mason & Rich, 2019), and the absence of systematic policies from schools (Fraillon et al., 2020). By incorporating coding concepts into mathematics

—a core school subject—these barriers can be effectively addressed (Dohn 2020; Popat & Starkey, 2019). This approach emphasizes the direct application of mathematical principles to understand and apply coding competencies (Dohn 2020; Popat & Starkey, 2019).

Two kinds of coding competencies can be developed through contexts of mathematics teaching and learning. First, specific coding skills, including specific CT skills, such as debugging, sequential skills, and abstraction thinking, can be enhanced. Among these crucial coding skills, abstraction plays an important role as it enables students to generalize from specific instances, focusing on relevant information while disregarding unnecessary details (Wing, 2006; Wing, 2011). A study conducted by Leonard et al., (2016) found that when designing mathematics related games, students' abstraction skills significantly improved. Specifically, the study showed that Grade 7th students demonstrated an enhance capability for abstraction by creating complex games. These games incorporated elements of cultures, procedural knowledges, and both incremental and goal-directed learning in their design. These results were aligned with research conducted by Stigberg and Stigberg (2020), who found that students' sequential skills would be improved, when introducing programming curriculum into primary mathematics instruction. The study was situated in Sweden, where programming had been integrated into mathematics curricula with the objectives to foster students' problem solving, logical thinking skills and motivate students to learn mathematics (Stigberg & Stigberg, 2020). Specifically, to improve students' sequential skills (i.e., the ability to organize ideas in the correct chronological order), the teachers designed an unplugged activity called a clapping chant. In the activity, the students worked in pairs, alternated counts to three, and gradually replaced "one" with a clap, "two" with a foot stamp, and "three" with finger snapping to master sequential

skills. In sum, these findings above emphasize the positive influence of mathematics contexts on the development of coding skills.

Second, within mathematics contexts, coding attitudes, such as confidence, motivation, self-efficacy can also be enhanced (Chan et al., 2023). For example, Dohn (2020) investigated whether the students' interest in coding would improve in lower secondary mathematics settings. Specifically, the researcher designed a course that included six consecutive mathematics lessons over three weeks, requiring students to use the learning resources titled "Programming in Math" to learn mathematics functions. A statistically significant increase in students' coding interest was observed after the mathematics lessons.

In summary, mathematics contexts not only facilitate the development of coding skills, such as sequential and abstraction skills, but also positively influence students' attitudes towards coding, including coding interest. This approach emphasizes the importance of mathematics, a core educational subject, in overcoming common barriers to coding education and demonstrates a reciprocal enrichment between mathematics learning and coding competencies.

2.3 An Alternate Framework to Support Integration

In the last section, I presented research showing two different ways in which coding and mathematics have been integrated. However, in these studies, CT was only partially achieved, if at all. Considering that CT is a specific form of coding, I will introduce a pedagogical framework developed by Kotsopoulos et al. (2017) that focuses on types of activities that can foster CT. The selection of this framework is based on its comprehensive structure and adaptability across educational levels. Moreover, the chosen model focuses on how to integrate CT across various

disciplines, suggesting its potential to support integration of coding with CT. In this section, I will detail this model.

Kotsopoulos et al. (2017) developed their framework based on the available literature on CT. It consists of four types of activities: unplugged, tinkering, making, and remixing. *Unplugged experiences* involve coding concepts without digital devices, such as computers, phones, and robotics. The idea for incorporating unplugged activities into the CT model stems from the desire to make CT more accessible to young children (Kotsopoulos et al., 2017). It is particularly beneficial for beginners, young students, or those with limited access to technology due to financial constraints (Lee & Junoh, 2019). *Tinkering* involves disassembling and making minor adjustments to existing objects. It encourages exploration through "what if" scenarios. The reason for integrating tinkering within instruction arises from the recognition of the natural curiosity and exploratory behavior that learners, especially children, exhibit towards understanding how things work (Kotsopoulos et al., 2017). Contrary to tinkering, which involves modifying or experimenting with existing objects, *making* is defined as the process of creating entirely new objects from scratch, utilizing a diverse combination of resources and tools (Papert, 1980). The decision to incorporate making within the CT model is driven by an emphasis on creation and innovation as central to learning and understanding technology and complex coding concepts (Kotsopoulos et al., 2017). Finally, *remixing* involves applying acquired knowledge to different contexts, such as real-life or financial scenarios, and across disciplines like literacy or arts. It extends beyond mere problem-solving within familiar contexts, advocating for the application of knowledge in novel settings (Kotsopoulos et al., 2017). And the decision to integrate remixing within the CT model is motivated by the recognition of adaptability, creativity, and cross-disciplinary learning as pivotal in the digital age (Kotsopoulos et al., 2017).

I used Kotsopoulos's pedagogical framework as a conceptual framework for my thesis for several reasons. First, the framework advocates for hands-on, experiential learning, and it encourages students to actively engage with coding and CT concepts through practical activities. Secondly, it aims to promote creativity and exploration in the learning process and foster a dynamic and engaging educational experience. Moreover, the framework aims to connect coding/CT education to real-world applications, demonstrating the practical relevance of coding/CT skills and reinforcing the transferability of knowledge from theories to practices. Additionally, the framework is tailored for teachers and practitioners, and it aims to provide them with a structured pedagogical framework for integrating coding into real practices. It shows a broad scope, encompassing various pedagogical approaches and activities to enhance the learning experiences. Based on all these attributes of the framework, this framework aligns with one of the study's objectives, which is to identify various coding learning opportunities created by lesson plan designers (Chapter 6).

2.4 Conclusion of Chapter 2

In this chapter, I first explored the relationships between coding and CT, where I regarded CT as a specific form of coding within the context of this study. Specifically, CT in this study extends beyond programming and is considered to encompass a range of skills, including design thinking, problem solving, etc. I then discussed two ways that coding and mathematics have been integrated in the research literature, as outlined in Themes 1 and 2. Specifically, Theme 1 demonstrated that within coding contexts, mathematics competencies could be improved, while Theme 2 illustrated that within mathematics contexts, coding competencies could be enhanced. Generally, these two themes offered a diverse range of perspectives on this

topic, highlighting the broad consensus on coding's importance, innovation, and utility in mathematics teaching and learning. Finally, I introduced a pedagogical framework with the potential to fully integrate CT into mathematics, further guiding my selection of an appropriate conceptual framework for this study. This literature review therefore established a solid foundation for my master's thesis. In the next chapter, I will present the conceptual framework used in the study.

Chapter 3: Conceptual Framework

In this chapter, I begin by explaining how I developed my conceptual framework. I then delve into two aspects related to it: (1) content knowledge in mathematics teaching and (2) curriculum analysis. Finally, I present my conceptual framework, which I have named *Coding Content Knowledge for Mathematics Teaching and Learning (CCKMTL)*.

3.1 Ideas Building Towards my Conceptual Framework

I construct my conceptual framework from two theoretical perspectives. The first perspective comes from the idea that coding is the content knowledge that teachers must understand and integrate into their teaching mathematics practices. This study thus adopts Ball et al.'s (2008) definition of content knowledge for teaching, which includes two main components: subject matter knowledge and pedagogical content knowledge. The second perspective stems from the consideration of how curricula can be analyzed. Consequently, this study adopts Posner's (2004) argument that curriculum analysis (CA) serves as a method to guide this study's analysis.

By combining these two perspectives, I have developed a conceptual framework that serves as a detailed guide for investigating how to integrate coding into mathematics curricula. In the next section (Section 3.1), I will discuss the first perspective, which relates to content knowledge.

3.2 Content Knowledge in Mathematics Teaching (Subject and Pedagogy)

This first perspective is based on Ball et al.'s (2008) definition of content knowledge for teaching, which includes two components: subject matter knowledge and pedagogical content knowledge. In this section, I first explain what subject matter knowledge entails and then discuss pedagogical content knowledge.

3.2.1 Subject Matter Knowledge

Subject matter knowledge (SMK) refers to a deep understanding of the content to be taught within a specific discipline (Shulman, 1986). It includes a comprehensive grasp of the concepts, theories, and principles and methodologies within the discipline (Ball et al., 2008).

In my master's thesis, SMK extends beyond traditional boundaries and includes two disciplines: mathematics and coding. This interdisciplinary approach recognizes that the skill sets, and conceptual understanding required for mathematics are now linked with coding competencies. Specifically, the Ministry of Education of Ontario (2020) acknowledges the importance of this integration and outlines the specific goals within the mathematics curricula that integrate coding across various grades (i.e., Grades 1 to 8), aiming to equip students with a comprehensive skill set for both mathematics and coding. Therefore, in my study, to possess SMK, educators must have a deep understanding of both disciplines.

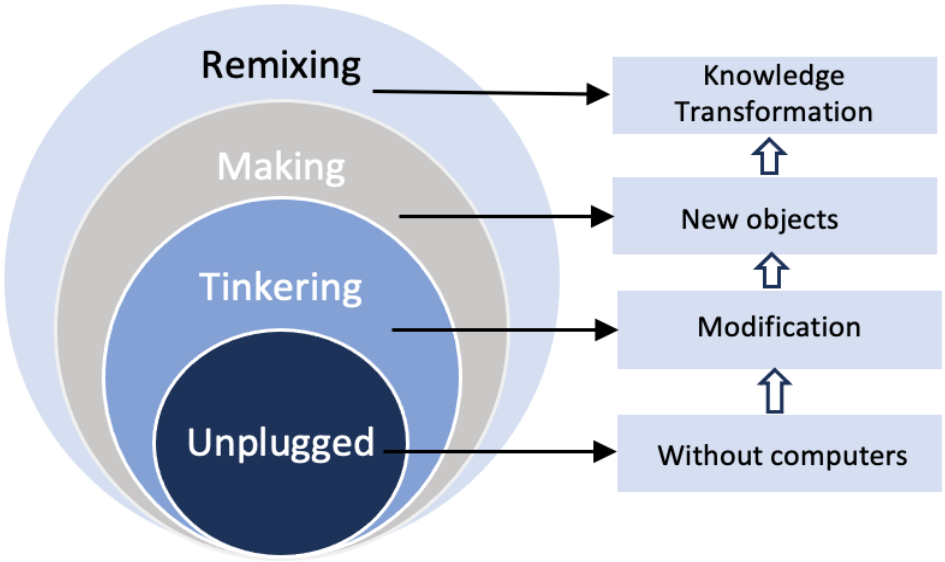
3.2.2 Pedagogical Content Knowledge

Pedagogical content knowledge (PCK) represents a blend of content expertise and pedagogical skills and enables teachers to tailor their instruction in ways that make the SMK accessible and engaging to learners (Ball et al., 2008). In my thesis, PCK is examined in the context of how lesson plan designers can adapt their instruction to complement the Ontario mathematics curricula and incorporate coding.

However, given the broad scope of pedagogical content knowledge as well as this study's focus on identifying the ways that coding was integrated into mathematics lessons, I have decided to incorporate the pedagogical framework developed by Kotsopoulos et al. (2017) into the PCK domain. This incorporation involved several steps. Firstly, I delineated the core components of the PCK domain – knowledge of content, knowledge of pedagogy, and the intersection where pedagogical strategies practitioners (e.g., lesson plan designers) could use to meet SMK. Similarly, I outlined the main elements of the pedagogical model developed by Kotsopoulos et al. (2017), focusing on its strategies, principles, and outcomes as they related to coding and mathematics. Secondly, I mapped the key elements of Kotsopoulos et al. (2017)'s model onto PCK domain. This involved aligning specific pedagogical activities from the model with PCK. The goal was to identify overlaps and areas where the model could fill gaps in traditional PCK regarding the coding integration into mathematics teaching and learning. Finally, I extracted the four components from Kotsopoulos et al. (2017)'s model (i.e., unplugged, tinkering, making, and remixing) to analyze the PCK related contents (i.e., in this study, it referred to the effective activities for coding integration within Ontario Math Support's lesson plans). Overall, this framework can provide a structured approach to analyzing and identifying effective strategies for integrating coding into the teaching and learning of mathematics. The framework applied is depicted below (see Figure 1), as previously discussed in section 2.3.

Figure 1

Pedagogical Framework for Coding (Kotsopoulos et al., 2017)



In sum, SMK and PCK are two fundamental aspects of content knowledge for understanding how to integrate coding into mathematics curricula. These two elements will guide my research as I identify the conceptualization of coding and the coding learning opportunities provided by lesson plan designers. In the next section, I will delve into the importance of curriculum analysis.

3.3 Curriculum Analysis Model

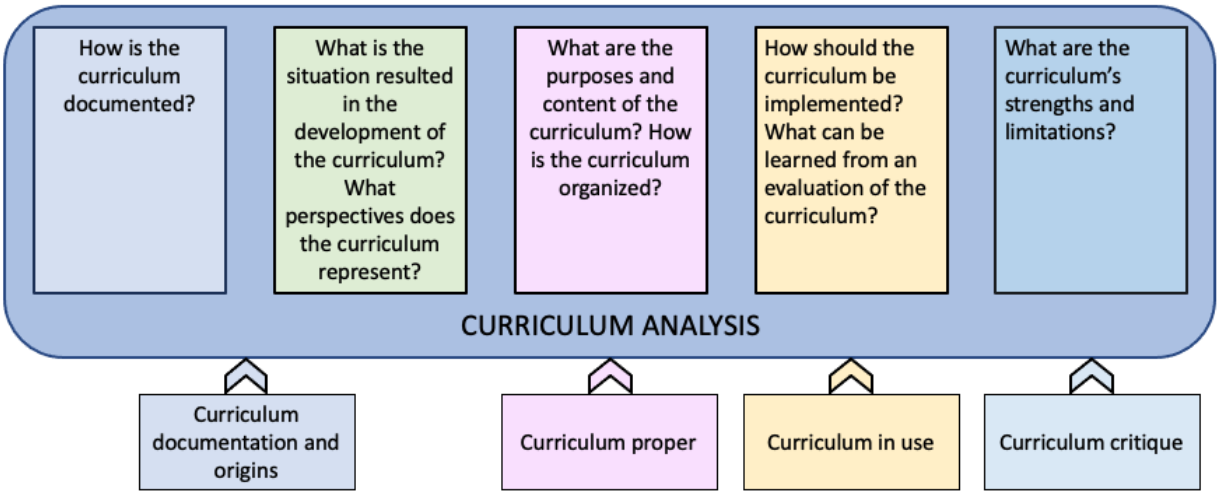
A curriculum is a comprehensive plan for an educational program that outlines the learning objectives, content, teaching strategies, and assessment tools, and resources required to facilitate effective learning (Young, 2014). It serves as a guide for both teachers and students by

providing a structural framework that details what needs to be taught, how it should be taught, and how learning could be assessed (Shkedi, 2009). Curricula (i.e., plural noun of curriculum) are designed to ensure that education is coherent and comprehensive, meeting the educational requirements of a school or educational system (Connelly & Connelly, 2012). Having established a foundational understanding of what constitutes a curriculum and its pivotal role in structing the educational experience, the focus now shifts to curriculum analysis.

Curriculum analysis (CA) evaluates how each part (i.e., curriculum) fits together in focus and coherence (Posner, 2004). In this master's thesis, I adopted Posner's definition of CA (2004) that dissects a curriculum into its individual elements, evaluates the coherence among these elements, identifies the foundational beliefs and ideas that influence its formation, and examines how these guiding principles impact the quality of educational experience. Specifically, there are four components within Posner's CA (2004): curriculum documentation and origin, curriculum proper, curriculum in use, and curriculum critique. Figure 2 presents the details of these components within CA.

Figure 2

Curriculum Analysis (Posner, 2004)



Briefly, *curriculum documentation and origin* refer to the fundamental documents that outline the curriculum's objectives, contents, teaching strategies, and even assessment tools. It also involves the historical contexts behind the curricula's development, including educational policies, theories, and societal needs (Posner, 2004). *Curriculum proper* refers to the official curricula used by educational institutions. It is what educators expect to follow and implement in their practice (Posner, 2004). *Curriculum in use* concentrates on the actual implementation of the curricula in practice. It involves how the teachers and educators interpret and deliver the curriculum contents to students (Posner, 2004). Finally, *curriculum critique* involves a critical analysis or evaluation of the curriculum from various perspectives. The process often leads to recommendations for curriculum improvement (Posner, 2004). In conclusion, CA is a multidimensional process through which the researcher seeks to understand the curriculum as a dynamic cultural artifact shaped by various educational, social, and ideological influences (Prentice, 1971). It aims to enhance the quality of curriculum design and implementation (Posner, 2004).

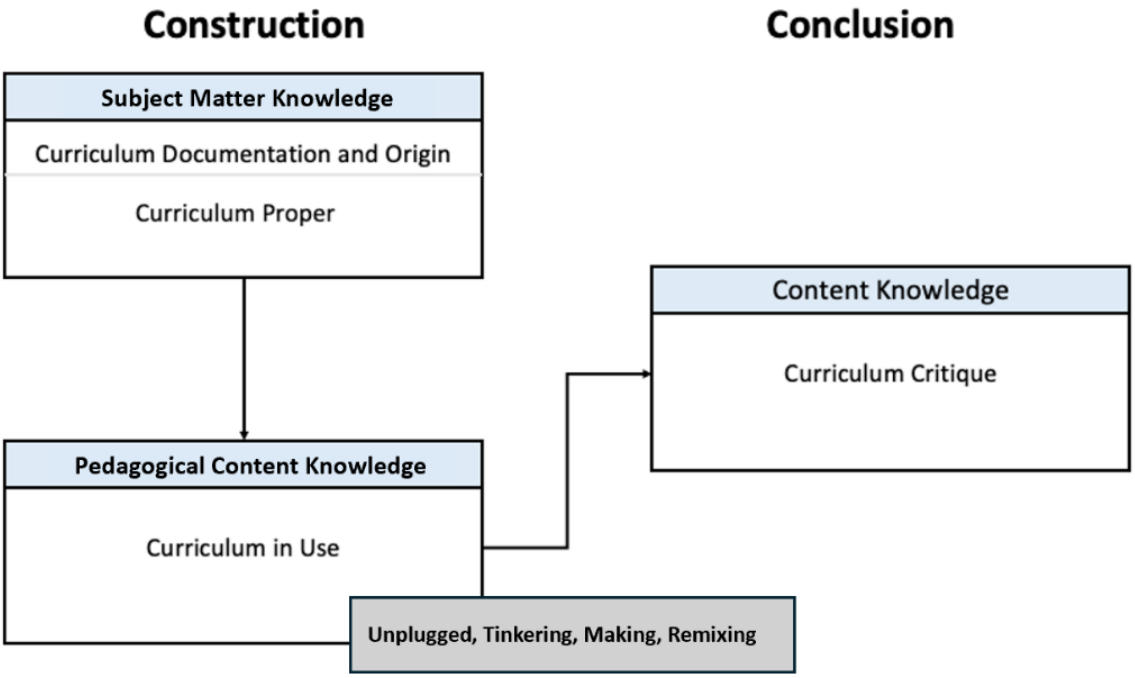
As we can see, these two perspectives (content knowledge for teaching and curriculum analysis) have the potential to help me understand (1) what kind of content knowledge educators should be equipped with when integrating coding into teaching and learning mathematics and (2) how to analyze curricula as a researcher. It leads to my next section where I integrate the two perspectives (content knowledge and curriculum analysis) to construct my conceptual framework.

3.4 Conceptual Framework

I build my conceptual framework (see Figure 3) based on the two perspectives discussed above: content knowledge (Ball et al., 2008) and curriculum analysis (Posner, 2004). I call this framework Coding Content Knowledge for Mathematics Teaching and Learning (CCKMTL).

Figure 3

Conceptual Framework



As shown in Figure 3, there are three aspects within my conceptual framework. In the first part of the framework, I focus on SMK, curriculum documentation and origin, and curriculum proper. This part aims to (1) analyze the curriculum documentation, examining how the curriculum is documented and presented and (2) investigate its origins, exploring what perspectives are embedded in the curricula and the rationale behind its development.

In the second part of the framework, the focus shifts to PCK, and curriculum in use. The aims of this part are to (1) investigate the application of coding within the context of mathematics education, understanding how coding concepts are integrated into Ontario Math Support's lesson plans and (2) analyze how coding can be applied to real-life situations, such as financial mathematics, mathematics in technology-rich environments, or ethics in mathematics. This consideration emphasizes the importance of learning opportunities provided by educators in

facilitating coding integration within mathematics contexts. Thus, I integrate a pedagogical framework (Kotsopoulos et al., 2017) within this part, which comprises four components: unplugged, tinkering, making, and remixing. These components are designed to provide a comprehensive view to understand different learning opportunities for coding within mathematics contexts.

The third part focuses on criticizing the coding integration into curricula from an overall perspective. The reason I use content knowledge is that it merges the analysis results from both SMK and PCK, thus providing a critical evaluation of the coding integration within Ontario mathematics curricula. The discussion of this part is detailed in Chapter 7, Section 7.4 limitations.

In conclusion, CCKMTL is designed to analyze the complex interplay between content knowledge for teaching, specifically SMK and PCK, and CA that supports coding integration into mathematics contexts. It serves not only as a tool for academic investigation but also as a guide for educators, curriculum designers, and policymakers striving to effectively integrate coding within mathematics curricula. In the next chapter, I will present my methodology for the study.

Chapter 4: Methodology

In the chapter, I first provide a rationale for choosing to conduct a qualitative research method for this study. I then explain why and how I chose case study approach to answer my first research question and a content analysis method to address my second research question.

4.1 Qualitative Research Design

In this study, I employed a qualitative research method (Fossey et al., 2002). Qualitative research design is a methodological approach used to explore social phenomena (Fossey et al., 2002). Unlike quantitative research, which focuses on numerical and statistical analysis to draw conclusions, qualitative research emphasizes the importance to understand the quality, meaning, and depth of the content (Fossey et al., 2002).

There are two reasons why I chose qualitative research for my study. Firstly, qualitative research allows me for conducting a deeper exploration of Ontario's curricula to understand the reasons why Ministry of Education of Ontario (2020) integrates coding into mathematics contexts. Secondly, qualitative research is ideal for studying phenomena within their natural settings (Merriam, 2022), allowing me to observe and interpret different coding learning opportunities created by Ontario lesson plan designers. Therefore, I could capture the nuance of coding integration within Ontario mathematics curricula.

Specifically, to answer the first research question (How was coding conceptualized in the Ontario elementary mathematics curricula?), I chose the case study methodology as the primary research approach. To answer the second research question (How was coding integrated into Ontario Math Support's lesson plans?), I chose content analysis method as the main approach.

First, I will present the details of the case study method and justify why this approach is appropriate for my study.

4.2 Data and Analysis for Research Question 1

4.2.1 Choice of Case Study Methodology

To answer the first question, I chose case study approach as the primary method. Case study consists of a detailed investigation and allows for data collected over different educational levels within a specific context (i.e., Ontario mathematics curricula) (Hartley, 2004). In this study, there are three cases identified within Ontario elementary mathematics curricula: primary elementary (Grades 1 to 3; Case 1), junior elementary (Grades 4 to 6; Case 2), and intermediate elementary (Grades 7 to 8; Case 3).

There are two reasons why I chose case approach. Firstly, case studies can allow for an in-depth exploration (Yin, 2003). Specifically, each case (i.e., Case 1: Grades 1 to 3; Case 2: Grades 4 to 6; Case 3: Grades 7 to 8) in this study allows for a focused examination of the curricula specific to that educational level. This targeted approach can highlight the nuanced insights into how coding is introduced and evolves as students progress. Secondly, by analyzing different cases within the same educational system, I can compare how coding is approached and conceptualized according to each educational level. Thus, I can identify patterns, discrepancies, and opportunities for integrating coding into mathematics contexts according to specific educational level (Dai et al., 2023).

4.2.2 Data Collection and Analysis

Official Ontario curricula (Ministry of Education of Ontario, 2020), including all the curricular documents published for coding integration within mathematics contexts from Grade 1 to Grade 8, were selected as data. I purposefully selected the curricular documents because (1) they were publicly available, (2) they were about coding integration within mathematics curricula, and (3) they were from Grades 1 to 8. Then, based on the educational levels suggested by the Ministry of Education of Ontario (Ontario, 2024), I categorized them into three individual cases: Case 1 for Grades 1 to 3 (primary elementary); Case 2 for Grades 4 to 6 (junior elementary); and Case 3 for Grades 7 to 8 (intermediate elementary). These three cases were categorized based on Ontario schooling systems which would be discussed in Chapter 5, section 5.1. Additionally, a coding scheme, including coding concepts and skills, was developed for the data analysis. There were two steps how I coded these cases. Firstly, I created a code book that included "coding concepts," "coding skills," "coding benefits," and "coding's connection to the previous / next educational level" to analyze "what coding is." Secondly, I used the open coding method to examine "how coding is integrated into official Ontario curricula". The code book will be presented in Chapter 5, section 5.2.

4.3 Data and Analysis for Research Question 2

4.3.1 Choice of Content Analysis Method

Regarding question two, I chose content analysis as the primary method. Content analysis is a research method used to systematically analyze the content of materials. It is a technique for

making replicable and valid references from texts to the contexts of their use. This method focuses on interpreting the meaning of content in context (Drisko & Maschi, 2016).

There are two reasons why I made this choice. Firstly, content analysis allowed me to interpret content within its broader social, cultural, and educational contexts. In this study, it is crucial for me to comprehend how coding was implemented by lesson plan designers within mathematics contexts. Secondly, content analysis has the capability to produce both qualitative and quantitative data. Specifically, in this research, it enabled me to not only derive meaningful themes and narratives for various coding learning opportunities, but also count and analyze the occurrences of specific coding learning opportunities (i.e., unplugged, tinkering, making, and remixing).

4.3.2 Data Collection and Analysis

Thirty-five lesson plans provided by Ontario Math Support were selected as data (https://ontariomath.support). Table 1 presents the lesson plans provided by the Ontario Math Support website (Ontario Math Support, 2024).

Table 1

Lesson Plans Analyzed in the Research

	Name of the lesson plans	Expectations for coding*
Grade 1	• Story characters on a grid; • Please direct me to my seat; • Let's play hockey; • Barnard commotion and motion; • Hen runs away from fox.	• "Solve problems and create computational representations of mathematical situation using coding concepts and skills (i.e., sequential events)" (Ontario, 2022, Grade 1 C3).

Grade 2	• Coding with coins; • Dancing to code; • Building number sentences; • Coding for the squirrel.	• "Solve problems and create computational representations of mathematical situation using coding concepts and skills (i.e., sequential, and concurrent events)" (Ontario, 2022, Grade 2 C3).
Grade 3	• Dance patterns; • Making change; • Coding a table of value.	• "Solve problems and create computational representations of mathematical situation using coding concepts and skills (i.e., sequential, concurrent, and repeated events); Identify, describe, extend, create, and make predictions about a variety of pattens, including those found in real-life contexts" (Ontario, 2022, Grade 3 C3).
Grade 4	• Coding the way; • Dancing our way to coding using patterning; • Coding on a coordinate grid; • Let's paint our classroom; • Scratch your probabilities.	• "Solve problems and create computational representations of mathematical situations by writing and executing code, including code that involves sequential, concurrent, repeating, and nested events" (Ontario, 2022, Grade 4 C3).
Grade 5	• Budgets and spreadsheets: Making plans; • Navigational coding; • Recalling math facts through coding; • Exploring patterns with fractions using scratch.	• "Solve problems and create computational representations of mathematical situations by writing and executing code, including code that involves conditional statements and other https://www.dcp.edu.gov.on.ca/en/"(Ontario, 2022, Grade 5 C3).
Grade 6	• Coding transformations; • Part1: which integer is greater? • Part2: Mystery Integer! • Start small – making informed; financial decisions; • Coding and composite numbers.	• "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" (Ontario, 2022, Grade 6 C3).

Grade 7	• Get the wolf to her pups; • Restaurant order app; • $20 or 20%; • Draw a maze;	• "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 subprogram and other control structures" (Ontario, 2022, Grade 7 C3).
Grade 8	• Coding and geometry; • Coding with Pythagorean theorem; • Coding YouTube	• "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" (Ontario, 2022, Grade 8 C3).

** These expectations are retrieved from the Ontario Ministry of Education (Ministry of Education of Ontario, 2020; Ontario, 2022)*

These lesson plans were purposefully chosen based on four criteria: (1) they were publicly available, (2) they were aligned with Ontario's coding curricula (Ministry of Education of Ontario, 2020), (3) they were within mathematics contexts, and (4) they were from Grades 1 to 8. A coding scheme based on Kotsopoulos et al.'s (2017) pedagogical framework was then developed to facilitate data collection. This scheme included four categories: unplugged coding learning opportunities, tinkering coding learning opportunities, making coding learning opportunities, and remixing coding learning opportunities. Subsequently, content analysis was conducted to identify these four components of coding learning opportunities within the selected lesson plans. After that, thematic analysis (Braun & Clarke, 2012) was used to identify the themes and sub-themes emerging from each coding learning opportunity. Thematic analysis is a qualitative research method used to identify, analyze, and report themes within the data. Specifically, the four coding learning opportunities mentioned above, which focused on pedagogical integration of coding into mathematics lesson plans, provided lenses through which

the data was viewed. They guided me to pay particular attention to themes and subthemes related to specific coding learning opportunities (i.e., unplugged, tinkering, making, and remixing). These four coding learning opportunities acted as scaffolds that supported the thematic structure, ensuring the themes were directly derived from the data. Table 2 presents the themes and subthemes emerging from the data analysis process.

Table 2

Themes and Subthemes that Emerged from Thematic Analysis

Coding learning opportunities	Themes	Subthemes
Unplugged	Kinesthetic	Body, Dance, Tools
	Oral	Introduction, Discussion
Tinkering	Code manipulation	--
	Interactive learning	--
Making	Generalization	--
	Digital Making	Scratch, Google related tools, Others
Remixing	Mobilizing knowledge	--
	Application in various contexts	--

Finally, I created narrative descriptions to display how each coding learning opportunity was demonstrated across all the lesson plans. The detailed code books will be presented in Chapter 6, section 6.1.

In summary, this study employed case study approach to answer the first research question, and content analysis to address the second question, ensuring a comprehensive examination of coding integration within teaching and learning of mathematics in Ontario curricula. The chosen methods allowed for an in-depth understanding of how coding was conceptualized (first research question) and implemented (second research question) across various educational levels (Grades 1 to 8). In the next chapter, I will present the results from the case study analysis.

Chapter 5: Result 1 Ontario Curricula

In this chapter, I present the results for my first research question: "How was coding conceptualized in the Ontario elementary mathematics curricula?". I first present a brief description of the Ontario education system and why the Ministry of Education in Ontario aims to integrate coding in the mathematics curricula. I then compare primary elementary, junior elementary, and intermediate elementary curriculum documents using case study approach to explain how the Ministry of Education in Ontario conceptualizes coding within their mathematics curricula.

5.1 Curriculum Documentation and Origins: Ontario Education

In Ontario, the Ministry of Education is responsible for overseeing all aspects of the province's public education system (Ontario, 2024). Specifically, the Ministry of Education in Ontario has a website that offers services and resources for various communities (Ontario, 2024). Specifically, for parents, it offers detailed information about associated fees, fundraising efforts, and ways to engage in their children's education (Ontario, 2024). For teachers, it informs teachers of professional development opportunities and supplies essential teaching tools (Ontario, 2024). Finally, for school boards, it provides guidance on matters such as education funding grants, collective agreements, and additional resources (Ontario, 2024).

Following this overview of the Ministry's roles, it is also important to understand the structure of the schooling system that the Ministry governs. Specifically, kindergarten is offered for children from the ages of four to six; however, it is not mandatory. School attendance is

compulsory in Ontario from the age of six to eighteen. Students attend elementary school from Grade 1 to Grade 8, which covers the age range from 6 to 13 years old. After completing eight years of elementary school, students enter secondary school from Grade 9 to Grade 12. Upon completion of Grade 12, they can obtain their high school diploma (Ontario, 2021). Table 3 provides a brief introduction to the schooling system in Ontario. Next, I will provide an overview of the aims of coding in the Ontario elementary curricula (Grades 1-8).

Table 3

Ontario Schooling System (Ontario, 2021)

School Systems	Level	Grade	Age
Elementary	Kindergarten	Junior Kindergarten	4 years old
		Senior Kindergarten	5
	Primary	1-3	6-8
	Junior	4-6	9-11
	Intermediate	7-8	12-13
Secondary (High)	N/A	9-12	14-18
University and College	College	College's Degree	18-21
	University	Bachelor's degree	18-22

5.1.1 Ontario Curricula: Aims of Coding

In the Ontario curricula, coding is regarded as a fundamental component in mathematics, specifically in algebra (Ministry of Education of Ontario, 2020). In early grades, students are introduced to understand basic coding concepts (i.e., sequences, loops). This foundational

knowledge is progressively expanded upon, leading to more advanced programming abilities by Grade 8. By this stage, students are expected to proficiently write and execute code that addresses complex problems and conveys mathematical ideas computationally (Ministry of Education of Ontario, 2022, p. 477).

The Ontario curricular documents provide several reasons for emphasizing coding in the mathematics curricula. Firstly, a key reason for emphasizing coding in the curricula is its potential to enhance students' understanding of mathematical concepts and principles. According to the curricula, engaging students in coding activities can deepen their comprehension of mathematics through practical applications and problem solving (Ministry of Education of Ontario, 2020). Calder and Rhodes (2021) supported this reasoning, highlighting the significance of coding and collaborative learning in facilitating students' mathematical conceptual understanding and problem-solving abilities.

In addition, coding in Ontario curricula reflects a broad educational goal to equip students with 21st century skills that are essential for success in various fields, including STEM (science, technology, engineering, and mathematics) (Kanbul & Uzunboylu, 2017). The Ministry of Education of Ontario (2020) proposed that integrating coding into educational practices was not only about learning to program, but also about fostering a set of skills and ways of thinking that were crucial for navigating and succeeding in the modern world. These (i.e., a set of skills) include critical thinking, digital literacy, problem solving, and collaborative learning (Ministry of Education of Ontario, 2020). These skills could prepare students for an evolving future, which is aligned with the research of Tuomi et al. (2018) who proposed that coding skills were becoming more and more important in our evolving society.

In summary, the curricular documents suggest that integrating coding into K-8 Ontario mathematics curricula serves multiple objectives. This integration not only aims to enhance students' mathematical understanding through practice but also aspires to prepare them for the evolving digital future (Ministry of Education of Ontario, 2020). The objective reflects the commitment of the Ontario Ministry of Education to develop well-rounded citizens who are equipped to adapt to a rapidly changing world. The next section will present three cases of coding concepts and practice within the educational system (i.e., primary elementary, junior elementary, and intermediate elementary).

5.2 Curriculum Proper: Teacher Support for Coding Integration in Mathematics

This section uses the case study approach to analyze and compare the primary elementary, junior elementary, and intermediate elementary Ontario curricula. For each of my three cases, I present two steps. First, I focus on the "what" aspect, examining the curricula's specific contents centered on coding. Second, I concentrate on the "how" aspect, examining the conceptualization of coding.

5.2.1 Case 1: Grade 1-3 Primary Elementary

In Case 1 (i.e., primary elementary, from Grades 1 to 3), I begin by addressing the "what," aspect. I then explain the "how," aspect.

5.2.1.1 What. Starting with what**,** the curricular documents in Case 1 use the term "coding skills." This highlights that students may not only passively learn coding concepts like "variable," "operations," and "codes," but also actively engage in conducting coding activities

(Resnick & Rusk, 2020). Specifically, Grade 1 students begin by learning to construct and change codes for sequential operations, thus understanding the step-by-step execution of tasks. By Grade 2, the complexity increases as students will be required to tackle codes for concurrent operations. In addition, they will learn to manage multiple actions simultaneously. In Grade 3, students will use loops, or repeats, to automate long processes or to condense processes that involve patterns and repeating steps. Overall, the Ontario curricular documents within Case 1 emphasize a gradual development of students' coding skills and indicate that within each grade, students are introduced to understand and master more complex coding concepts.

Furthermore, beyond the technical skills, the curriculum is structured to offer students a range of benefits that extend into academic development. Firstly, they provide students with opportunities to improve mathematical understanding. Specifically, the logical structure of coding directly complements mathematical concepts, thus reinforcing skills such as pattern recognition, sequencing skills, and algorithmic thinking (Ontario, 2020). Secondly, coding at early ages promotes the development of social skills, such as communication (Ontario, 2020). Specifically, students at early ages are encouraged to describe how changes to code can affect the outcomes. Table 4 presents the coding concepts and skills expected for primary elementary students in Ontario.

Table 4

Coding Concepts and Skills for Primary Elementary

Grades	Coding Skills		Coding Concepts
Grade 1	"Solve problems and create computational representations of mathematical situations by writing and executing code, including code that involves sequential events" (Ontario, 2022, Grade 1 C3.1).	"Read and alter existing code, including code that involves sequential events, and describe how changes to the code affect the outcomes" (Ontario, 2022, Grade 1 C3.2).	"Sequential events: a set of instructions carried out one after another, usually top to bottom or left to right on a screen (*Note: see also concurrent events)"* (Ontario, 2022, Grade 1 C3.1).
Grade 2	"Solve problems and create computational representations of mathematical situations by writing and executing code, including code that involves sequential and concurrent events" (Ontario, 2022, Grade 2 C3.1).	"Read and alter existing code, including code that involves sequential and concurrent events, and describe how changes to the code affect the outcomes (Ontario, 2022, Grade 2" C3.2).	"Concurrent events: Two or more events that occur at the same time. *(Note: see also sequential events)"* (Ontario, 2022, Grade 2 C3.1).
Grade 3	"Solve problems and create computational representations of mathematical situations by writing and executing code, including code that involves sequential, concurrent, and repeating events (Ontario, 2022, Grade 3" C3.1).	"Read and alter existing code, including code that involves sequential concurrent, and repeating events, and describe how changes to the code affect the outcomes" (Ontario, 2022, Grade 3 C3.2)	"Repeating events: Something that happens over and over again. In coding, loops are used to repeat instructions. *(Note: See also loop)"* (Ontario, 2022, Grade 3 C3.1)

5.2.1.2 How. Coding in the case of primary elementary is introduced in a simplified form, often through visual languages. Visual languages, in this context of coding integration, especially for Grades 1 to 3 students, refer to using graphical representations (such as icons, symbols, bodies, or blocks) to convey programming concepts (Strawgacker & Bers, 2019). This approach might make it easier for the students who may not yet have the reading or typing skills necessary for traditional text-based programming languages (Noone & Mooney, 2018).

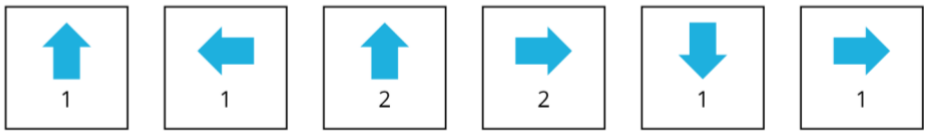
An example provided for Grade 1 students is shown in Figure 4, where they can explore mathematical situations through physical activities (i.e., a kind of visual language) that simulate coding logic. Specifically, in the example, the section called "code/instruction to model movements from one location to another" shows a sequence that corresponds to directional movements the student might use in a real block-based coding environment.

Figure 4

An Example from the Grade 1 Ontario Curriculum (Ontario, 2022)

Examples

- **possible mathematical situations:**
  - moving from one location to another
  - representing addition and subtraction of whole numbers
  - engaging in patterning using actions and colours
  - developing spatial reasoning
- **code/instructions to model movement from one location to another:**
  - up 1, left 1, up 2, right 2, down 1, right 1:

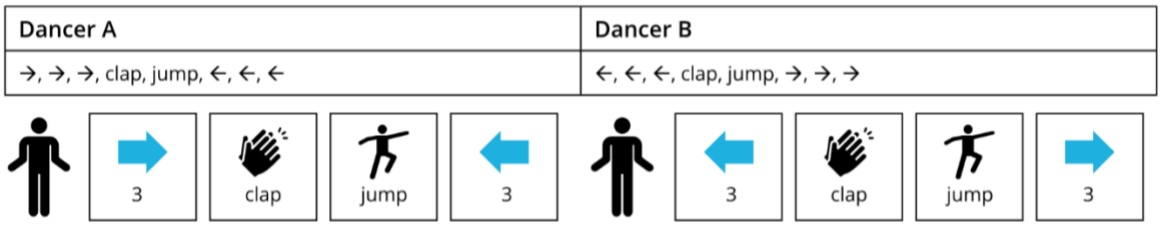


In Grade 2, it is also observed that coding is presented through physical activities. Figure 5 shows an example provided in the curriculum for Grade 2 where the students are engaged in a task: Working in pairs, they play the role of two dancers with the objective of synchronizing their dance movements. During the learning process, they learn the coding concept "concurrent events."

Figure 5

An Example from the Grade 2 Ontario Curriculum (Ontario, 2022)

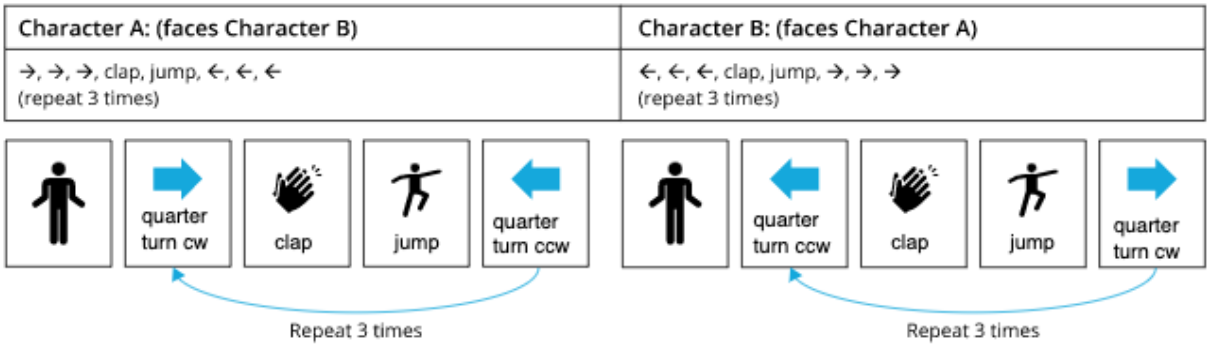
\



In Grade 3, there is a similar task provided as an example. However, compared to the one in Grade 2, this task, presented in Figure 6, expects students to understand what "loops" are by repeating the movement.

Figure 6

An Example from the Grade 3 Ontario Curriculum (Ontario, 2020)



All of the example tasks illustrated in Figures 4 – 6 can be regarded as unplugged activities. They emphasize the importance of teaching and learning coding without the use of digital devices (Battal et al., 2021). Through such activities, students may learn foundational coding concepts and logical sequencing in a tangible and kinesthetic manner, thus enhancing their understanding of coding principles through physical enactment (Lengel & Kuczale, 2010).

In summary, as students progress through the early stage of coding (primary elementary, Grades 1 – 3), the curricula suggest that students should gain coding skills and also develop a foundational understanding of using tools (e.g., unplugged activities) for creativity and problem solving. Moreover, this initial exposure lays the foundation for a more in-depth exploration of coding and its applications in later grades (i.e., Grades 4 – 8). Next, I will discuss Case 2, focusing on curricula in junior elementary.

5.2.2 Case 2: Grade 4-6 Junior Elementary

In Case 2 (i.e., junior elementary, from Grades 4 to 6), my discussion will start with the "what" aspect, followed by an explanation of the "how" aspect.

5.2.2.1 What. The curricula for junior elementary students include three coding concepts: nested events, conditional statements, and control structures. Specifically, in Grade 4, students will work with codes that include nested events. In Grade 5, the focus shifts to integrating both conditional statements and control structures. The curricula aim to enable students to perform actions based on specific conditions. In Grade 6, the curricula aim to refine students' coding efficiency, expecting them to achieve tasks using fewer instructions (Ministry of Education of Ontario, 2020).

In addition, these grade levels represent a continuation and expansion of the coding and skills acquired from primary elementary (Case 1) levels. For example, Grade 4 students learn about nested events through activities like creating repeating patterns with the rotation of a square, building upon the concept of repeating events introduced in Grade 3. Table 5 presents the coding concepts and skills expected for junior elementary students in Ontario.

Table 5

Coding Concepts and Skills for Junior Elementary

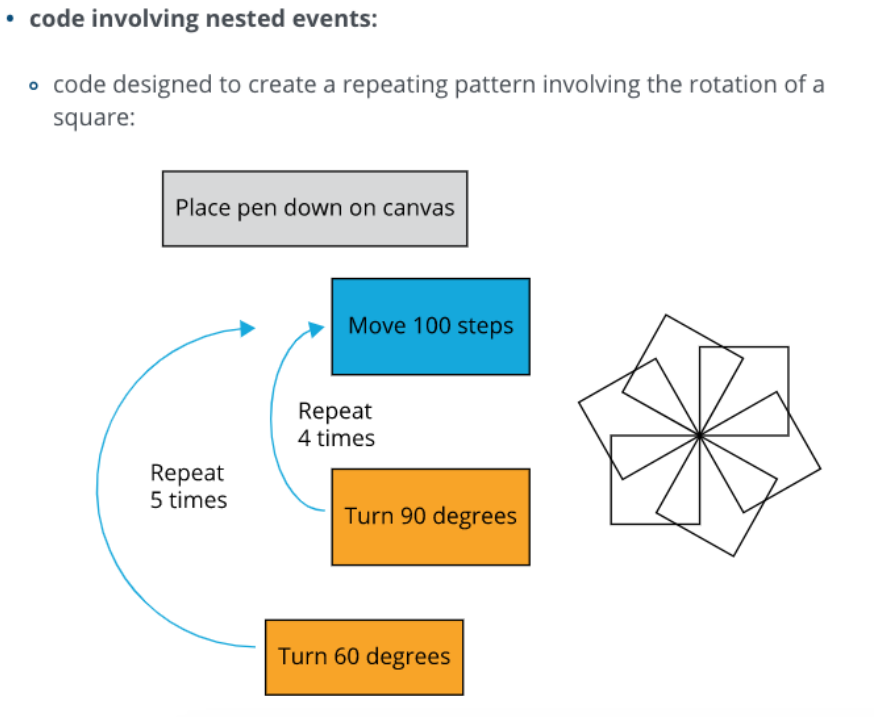
Grades	Coding Skills		Coding Concepts
Grade 4	"Solve problems and create computational representations of mathematical situations by writing and executing code, including code that involves sequential, concurrent, repeating, and nested events" (Ontario, 2022, Grade 4 C3.1).	"Read and alter existing code, including code that involves sequential, concurrent, repeating, and nested events, and describe how changes to the code affect the outcomes" (Ontario, 2022, Grade 4 C3.2).	"Nested events: Control structures that are placed inside other control structure; for example, loops occurring inside other loops, or a conditional statement being evaluated inside a loop *(Note: see also control structure)*" (Ontario, 2022, Grade 4 C3.1).
Grade 5	"Solve problems and create computational representations of mathematical situations by writing and executing code, including code that involves conditional statements and other control structures" (Ontario, 2022, Grade 5 C3.1).	"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" (Ontario, 2022, Grade 5 C3.2).	"Conditional statement: a type of coding instruction used to compare values and express and make decisions. A conditional statement tells a program to execute an action depending on whether a condition is true or false. It is often represented as an *if-then* or *if-then-else* statement" (Ontario, 2022, Grade 5 C3.1). "Control structures: a line or block of code that influences the order in which other code is executed. Control structures affect the flow of the program and include sequencing lines of code, repeating lines of code (loops), or selection to execute or not execute specific lines of code (conditional statements). Sequence, repetition, and selection are all control structures. *(Note: see also execute)*" (Ontario, 2022, Grade 5 C3.1).
Grade 6	"Solve problems and create computational representations of	"Read and alter existing code, including code that involves conditional	"Efficient code: code that uses the lowest number of instructions to accomplish a task, thereby minimizing storage space and execution time" (Ontario, 2022, Grade 6 C3.1).

	mathematical situations by writing and executing efficient code, including code that involves conditional statements and other control structures" (Ontario, 2022, Grade 6 C3.1).	statements and other control structures, and describe how changes to the code affect the outcomes and efficiency of the code" (Ontario, 2022, Grade 6 C3.2).	

5.2.2.2 How. Case 2 highlights coding as contexts for enhancing students' mathematics skills. For example, Figure 7 presents an example from Grade 4 (Ministry of Education of Ontario, 2020), where students are required to generate a repeating pattern through the rotation of a square. This coding activity involves multiple mathematics concepts, such as spatial reasoning (e.g., moving horizontally and vertically between locations), geometry (identifying shapes like squares and triangles), arithmetic operations (addition, subtraction, multiplication, and division of whole numbers), and the concept of repeating patterns in geometry. This approach reinforces the findings of my literature review, particularly Theme 2, suggesting that coding contexts have the potential to develop students' mathematics skills.

Figure 7

An Example from the Grade 4 Ontario Curriculum (Ontario, 2022)



In Grade 5, using coding contexts to improve students' specific mathematics skills can also be observed. Figure 8 showcases an example where the students are expected to improve their number sense (i.e., a mathematics skill) when they are learning the coding concept "conditional statements".

Figure 8

An Example from the Grade 5 Ontario Curriculum (Ontario, 2022)

- **mathematical situations involving conditional statements:**
  - comparing budgets to actual spending
  - comparing perimeters or areas of shapes
  - running probability simulations
  - classifying angles
  - solving inequalities, such as code that would determine who can ride the roller coaster at an amusement park:

```
If height >= 91 and height <= 102, then print "You
can ride!"
```

- determining the minimum and maximum values from a set of numbers
- programming a "Guess that Number" game:

```
If guess > actualNumber, then print "Sorry, your
guess was too high."
    else if guess < actualNumber, then print
    "Sorry, your guess was too low."
    else print "Yes! Your guess was correct."
```

- programming a rock, paper, scissors game
- **code involving conditional statements:**

"If then" statements:

```
If Area A == Area B
    Triangle A has the same area as Triangle B.
```

"If then else" statements:

```
If Area A == Area B
    Triangle A has the same area as Triangle B.
Else if Area A > Area B
    Triangle A's area is greater than Triangle B's
    area.
Else if Area A < Area B
    Triangle A's area is less than Triangle B's
    area.
```

A similar example is provided in the curricula from Grade 6. Figure 9 presents an example from Grade 6 where the students are expected to improve their calculation skills and logical thinking while grasping the coding concept "pseudocode."

Figure 9

An Example from the Grade 6 Ontario Curriculum (Ontario, 2022)

Provide students with code that needs to be altered and/or expanded upon. For example, the following pseudocode can be used to simulate flipping a coin twice, storing the occurrences, and calculating the percentage for each result. Have students alter this code to simulate a greater number of flips while keeping the code efficient.

heads = 0
tails = 0
headsPercent = 0
tailsPercent = 0
coin1 – random number 0 and 1
if **coin1** == 0 then
heads = **heads** + 1
else
tails = **tails** + 1
headsPercent = (**heads** / 2) * 100
tailsPercent = (**tails** / 2) * 100
output "Number of heads: ", **heads**
output "% heads: ", **headsPercent**
output "Number of tails: ", **tails**
output "% tails: ", **tailsPercent**

Note:

- Assignment statements assign a value to a variable and use a single equal sign (=), while comparison statements are used to compare two values and use a double equal sign (==) for equal to, < for less than, > for greater than, <= for less than or equal to, and >= for greater than or equal to.
- Pseudocode does not represent a specific programming language. It can be adapted to work with a variety of programming languages and/or environments.

In summary, coding in Case 2 is not only a tool for enhancing technical skills but also a means for deepening students' specific mathematical skills. It is expected to be a bridge between

primary elementary and intermediate elementary. Next, I will discuss Case 3, intermediate elementary.

5.2.3 Case 3: Grade 7–8 Intermediate Elementary

In Case 3, I will start by addressing the "what" aspect and I will then explain the "how" aspect.

5.2.3.1 What. At the intermediate level, two coding concepts are introduced in Grade 7: defined count and subprograms. In contrast, Grade 8 does not introduce new coding concepts to students. Instead, students are encouraged to reflect on the coding concepts and skills they have acquired previously and explore how coding can be utilized to manipulate data, which will facilitate more informed decision-making. According to the curricula, this approach aims to provide students with opportunities to (1) develop their data analytical skills and (2) apply coding from mathematics to other disciplines (e.g., data analysis). Table 6 presents the coding concepts and skills expected for intermediate elementary students in Ontario.

Table 6

Coding Concepts and Skills for Intermediate Elementary

Grades	Coding Skills		Coding Concepts
Grade 7	"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 subprogram and other control structures" (Ontario, 2022, Grade 7 C3.1).	"Read and alter existing code, including code that involves events influenced by a defined count and/or subprogram and other control structures, and describe how changes to the code affect the outcomes and the efficiency of the code" (Ontario, 2022, Grade 7 C3.2).	"Defined count: In coding, the number of times instructions are repeated based on a predefined value or until a condition has been met" (Ontario, 2022, Grade 7 C3). "Subprogram: a small set of instructions for completing one small task. Subprograms can be combined in a main program to accomplish a large task using small steps" (Ontario, 2022, Grade 7 C3).
Grade 8	"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" (Ontario, 2022, Grade 8 C3.1).	"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" (Ontario, 2022, Grade 8 C3.2).	"Flow chart: a type of diagram that shows the sequence of steps involved in performing an algorithm. In coding, specific symbols are used to indicate different control structures in the algorithm (e.g., an input is written in a parallelogram; a process is written using a rectangle)" (Ontario, 2022, Grade 8 C3).

5.2.3.2 How. At the intermediate elementary stage, according to the curricula, the students first need to consolidate their acquired knowledge. This consolidation is observed in the Grade 7 documents, where they indicate that students are required to review the coding concepts learned from previous grades, including sequential events, concurrent events, nested events, etc. Figure 10 presents an example from Grade 7 where the students are required to use the coding concepts learned from previous grades, such as the loops, conditions to create a shrinking pattern.

Figure 10

An Example from the Grade 7 Ontario Curriculum (Ontario, 2022)

```
termValue = 200
termNum = 0
repeat until termValue <= −100
    termValue = termValue − 50
    termNum = termNum + 1
    output termNum
    output termValue
output "There are ", termNum, " terms in this shrinking pattern."
```

In contrast to Grade 7, coding is transdisciplinary at Grade 8. By that stage, students are expected to apply coding across various disciplines, such as mathematics, finance, data analysis, literacy, etc. Figure 11 presents an example from Grade 8 where the students are expected to decide how to get a larger rectangular area of a school yard. Specifically, in this example, the

task of determining how to obtain a larger rectangular area of a schoolyard integrates coding with mathematics through the application of area calculation, finance through budget management for the project, data analysis in evaluating different configurations, and literacy in presenting the solution. This showcases how coding serves as a bridge across disciplines, enabling students to apply CT, problem solving, and mathematics skills in diverse contexts (i.e., finance, data analysis, literacy).

Figure 11

An Example from Grade 8 Ontario Curricula (Ontario, 2022)

The following subprogram determines how many items might be picked up in a large rectangular area of a schoolyard, and how long it would take, based on values input by students picking up items from a smaller rectangular area:

```
subprogram cleanUpCalculations
    smallArea = smallLength * smallWidth
    largeArea = largeLength * largeWidth
    scaleValue = largeArea / smallArea
    largeItems = smallItems * scaleValue
    largeMinutes = smallMinutes * scaleValue
    output "Small cleanup area (m2): ", smallArea
    output "Large cleanup area (m2): ", LargeArea
    output "Scale value: ", scaleValue
    output "Expected number of items in the large cleanup area: ", largeItems
    output "Expected minutes required to clean-up the large area: ", largeMinutes
```

In conclusion, in the intermediate elementary curricula, students are expected to consolidate coding concepts acquired in previous grades and learn to apply coding across various

contexts (e.g., mathematics, finance, literacy, data analysis), thus enhancing their analytical and mathematical abilities.

5.2.4 Summary of the Three Cases

In summary, these three cases present how the integration of coding with mathematics evolves from a foundational and simplified approach at the primary elementary level to a more advanced, transdisciplinary, and practically applied approach at the junior and intermediate elementary levels. This evolution stems from introducing coding as a visual tool (i.e., use coding contexts that incorporate visual languages, such as shapes, icons and even body movements to illustrate mathematical concepts and principles) for mathematics problem solving. It has the potential to foster students' creativity and practical understanding in complex, real-world scenarios across various disciplines. As students navigate diverse mathematical situations, the curricula are designed to encourage students to build computational skills and also develop a deeper comprehension for the role of coding in understanding mathematics and solving problems across various disciplines beyond mathematics. In the next section, I will present the summary of this chapter.

5.3 Conclusion of Chapter 5

In conclusion, this chapter elucidated the aims of coding as integrated into the Ontario mathematics curricula. It began by detailing the goals for coding integration set by the Ontario Ministry of Education, which included fostering CT, and preparing students for a tech-rich

society. I then presented three cases that examined how coding was conceptualized and integrated into different educational levels within Ontario's curriculum. The case studies spanned through the primary, junior, and intermediate levels of elementary education, providing a comprehensive analysis of what coding was, and how coding was used and applied across different levels. The findings from the case studies offered valuable insights into the effectiveness of the curricula in achieving its intended aims.

Therefore, building on the insights gained from this detailed examination, the next chapter (chapter 6) will delve into the practical application of the Ontario curricula. It will focus on how lesson plan designers craft coding learning opportunities within mathematics contexts, effectively bridging theoretical coding knowledge with practical coding skills.

Chapter 6: Result 2 Ontario Curriculum in Use

In this chapter, I present my results for my second research question about how Ontario Math Support's lesson plan designers provided coding learning opportunities (i.e., curriculum-in-use). To achieve this goal, I applied a pedagogical framework created by Kotsopoulos et al. (2017) to identify the coding learning opportunities within 35 lesson plans provided by the Ontario Math Support website.

6.1 Ontario Curriculum in Use

By adopting Kotsopoulos et al.'s (2017) pedagogical framework, I developed a codebook that included four components: unplugged, tinkering, making, and remixing. I then coded each lesson plan (n = 35) according to these components. Table 7 provides a summary of the coding opportunities I have identified from those 35 lesson plans. Specifically, the table's first column lists the grade levels, while the top row presents the four components.

Table 7

Summary of Coding Learning Opportunities in Lesson Plans

Grade	Name	Unplugged	Tinkering	Making	Remixing
Grade 1	Story characters on a grid	Physically move; hands on activity	Physically position through grid	---	---
	Please direct me to my seat	The entire lesson is unplugged coding	Iterative process of navigating the grid	Creation of grids and obstacles	---

	Let's play hockey	Physically movements; storytelling	Using coding cards to create scenarios	---	Transfer the knowledge to other contexts
	Barnard commotion and motion	Warm-up activities; physical movements	Pseudocode interpretation and movements	Students craft their own sequences of movements	Transfer the knowledge to other contexts
	Hen runs away from fox	Discussion and conceptualization of animation	Experiment with code	Create new things on a visual coding platform: Scratch	---
Total	5 Lesson plans	5/5	5/5	3/5	2/5
Grade	Name	Unplugged	Tinkering	Making	Remixing
Grade 2	Coding with coins	Physically move around a 10x10 grid	Write codes using a sequence of actions	Create codes on grip paper	---
	Dancing to code	Dancing, arrange cards	Modify dance sequence	Create own dance sequence	---
	Building number sentences	Use physical materials like 100's chart, counter, and sticky notes	Adjust number sentences and corresponding codes	Create their own number sentences	---
	Coding for squirrel	Students are encouraged to use mathematical tools	Create and modify their solutions	Create a map of forest on large sheets of grid paper	Lesson structure moves from adding numbers to sharing quantities
Total	4 Lesson plans	4/4	4/4	4/4	1/4
Grade	Name	Unplugged	Tinkering	Making	Remixing
Grade 3	Dance patterns	Dance moves	Use dance elements to expand the movement vocabulary with patterns	Create dance phrases	How original code was altered and resulted in new patterns

	Making change	Pre-existing code	Make changes using coins	Create a new version of code	---
	Coding a table of value	---	Alter code	Create new patterns	Real-life contexts
Total	3 Lesson plans	2/3	3/3	3/3	2/3
Grade	Name	Unplugged	Tinkering	Making	Remixing
Grade 4	Coding the way	Physical maps	Modify codes to navigate through the grid	Construct their own maps	Guide a partner through a grid using a set of oral coded translations
	Dancing our way to coding using patterning	Dance and physical movements	Create and adapt their dances	Work together to create repeating dance	Learn dances from other people's codes
	Coding on a coordinate gride	Paly a version of Battleship	Modify the game by arranging letters	Scratch	Real-world scenarios, like GPS
	Let's paint our classroom	Use concrete materials to understand area calculation;	Modify code sequences to determine the paint needed for wall	Scratch	---
	Scratch your probabilities	Discussion	Adjust codes	Scratch	Apply code to different probability contexts
Total	5 Lesson plans	5/5	5/5	5/5	4/5
Grade	Name	Unplugged	Tinkering	Making	Remixing
Grade 5	Budgets and spreadsheets: making plans	---	Explore the spreadsheets by clicking on cells to see the coding behind calculations	Create budget spreadsheets	Budgeting scenarios

	Navigational coding	Using coordinate plane	"If/then" statement	Create new maps; plugged in activities: Google Map	Different contexts: draw neighbourhood
	Recalling math facts through coding	---	Tinker existing codes to observe the relationship between multiplication facts and division	Create pixel	---
	Exploring patterns with fractions using scratch	---	Modify Scratch codes	Scratch	Connect to real-life scenarios
Total	4 Lesson plans	1/4	4/4	4/4	3/4
Grade	Name	Unplugged	Tinkering	Making	Remixing
Grade 6	Coding transformation	Use graph paper programming	Determine if the instructions will successfully transfer an image	---	---
	Part1: which integer is greater	Role plays between teachers and students	Modify codes in Scratch	Scratch	---
	Part2: mystery integer!	Introduction and direct instruction for pseudocode	Translate pseudocode to Scratch	Scratch	Apply skills in new projects
	Start small: making informed; financial decisions	Discussion	---	Google sheets	Financial contexts
	Coding and composite number	Introduction	Translate pseudocode to Scratch	Scratch	Apply to new contexts
Total	5 Lesson plans	5/5	4/5	4/5	3/5
Grade	Name	Unplugged	Tinkering	Making	Remixing

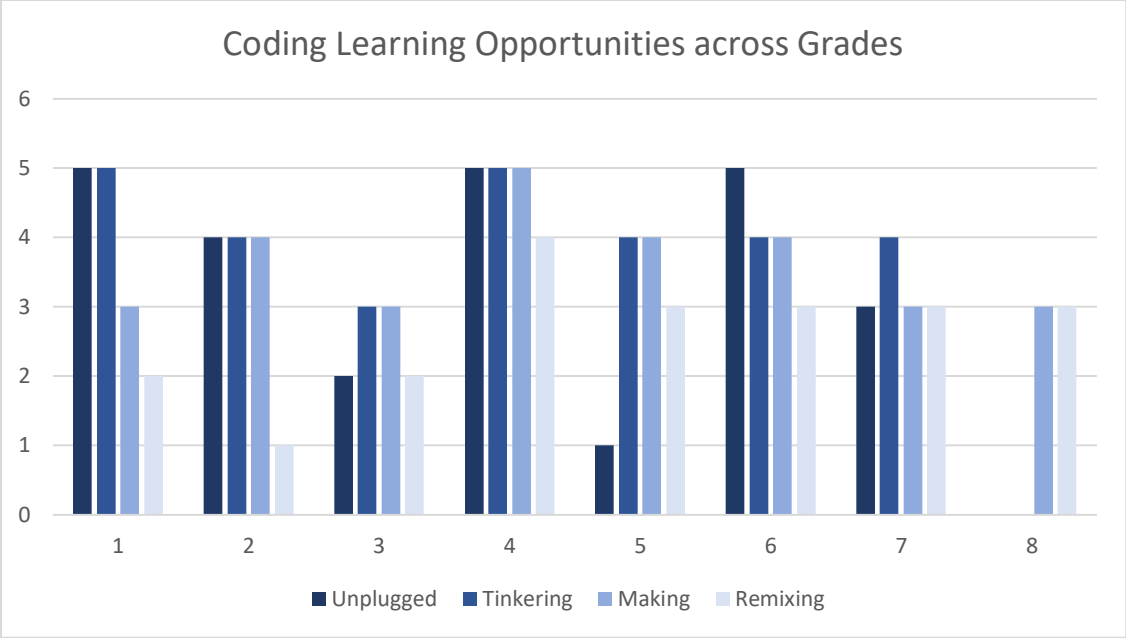
Grade 7	Get the wolf to her pups	Cartesian plane	Alter instructions	---	---
	Restaurant order app	---	Adding features like automatic calculation of total cost	Restaurant order app	The new context is the development of restaurant ordering app
	$20 or 20%	Discussion and math talks	Modify existing codes	A discount app	Real-world scenarios
	Draw a maze	Discussion	Making changes through Scratch	Scratch	Historical mathematics
Total	4 Lesson plans	3/4	4/4	3/4	3/4
Grade	Name	Unplugged	Tinkering	Making	Remixing
Grade 8	Coding and geometry	---	---	Scratch	Apply knowledge of geometric shapes to code transformation in Scratch
	Coding with Pythagorean theorem	---	---	Scratch	Apply their understanding of Pythagorean Theorem to different contexts
	Coding YouTube	---	---	YouTube monetization	Use "if/then" to create algorithms
Total	3 Lesson plans	0/3	0/3	3/3	3/3
Total for all grades	35 Lesson plans	25/35	28/35	28/35	21/35

In addition, Figure 12 shows the distribution of four coding learning opportunities across 35 lesson plans from Grades 1 to 8: unplugged, tinkering, making, and remixing. Four trends are noticeable in the table. First, unplugged opportunities are most prominent in the earliest grades, especially in Grade 1 and Grade 4, which show the highest levels of unplugged opportunities. There is a noticeable decline as the grade levels increase, with the lowest presence in Grade 7 and none in Grade 8. Second, tinkering opportunities are consistently present across all grade levels but are most prominent in Grade 1. Unlike unplugged, tinkering does not show a significant decline and remains present through Grade 8. Third, making opportunities are introduced at a lower frequency than the unplugged opportunities in the early grades but maintain a consistent presence across all grades. The peak for making opportunities is found at Grade 1 and Grade 4, after which there is a slight reduction (i.e., Grades 1 to 2, there is a decline; from Grades 2 to 4, there is an increase; after Grades 4, there is a decline), but presence remains stable in the higher grades. Finally, remixing has the lowest overall presence among four opportunities. It begins with a modest representation in Grade 1 and continues with a low but steady presence up to Grade 8.

In the following sections, I will further unpack each of the four learning opportunities. I will first explore unplugged coding learning opportunities in the next section.

Figure 12

Bar Charts for Coding Opportunities in Lesson Plans



Note. In this figure, the vertical axis (1-8) represents grades 1 through 8, while the horizontal axis (0-20) indicates the number of lesson plans included. Each lesson plan is counted once per coding learning opportunity. For example, a lesson plan that comprises several unplugged activities is coded once under the unplugged learning opportunity category.

6.1.1 Unplugged for Curriculum in Use

As described in earlier chapters, unplugged activities emphasize active, physical engagement to teach and learn coding within mathematics. I identified 25 lesson plans that created unplugged opportunities to teach and learn coding. As shown in Table 8, two themes emerged from those unplugged opportunities: (1) unplugged is kinesthetic (n = 18) and (2) unplugged is oral (n = 7). This section is structured around these two main themes, each with its own set of sub-themes. The first theme, kinesthetics, has three sub themes: body, dance, and tools. The second theme, oral, is divided into two sub themes: introduction and discussion.

Table 8

Unplugged for Curriculum in Use

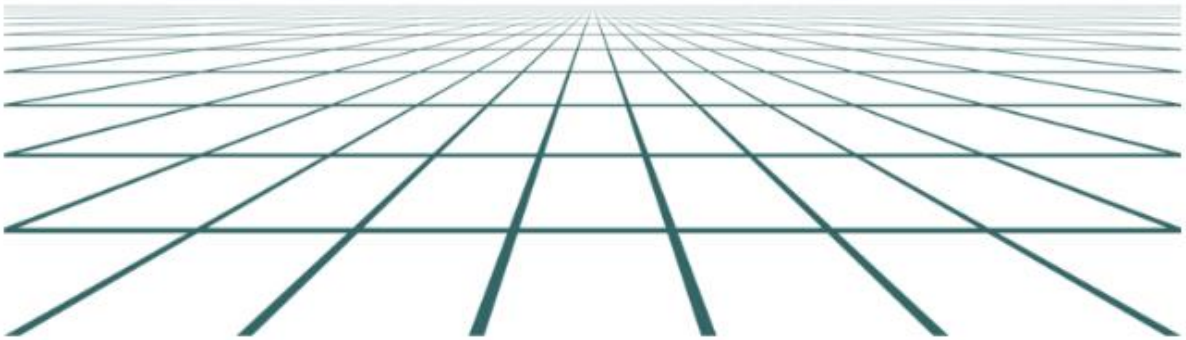
Codes	Sub-code	Explanation	Lesson Plans
Kinesthetic	Body	Students may use their body movements, hand signals, or facial expressions to represent different coding concepts or algorithm.	Story characters on a grid (G1) Please direct me to my seat (G1) Let's play hockey (G1) Barnard commotion and motion(G1) Coding with coins (G2) Making change (G3) Part1: which integer is greater (G6)
	Dance	Dancing can be used to represent code structure and flow. Dancing not only makes the process of learning coding fun but also helps in memorizing and understanding the sequence and structure.	Dancing to code (G2) Dance patterns (G3) Dancing our way to coding using patterning (G4)
	Tools	Tools refer to any physical objects or aids that facilitate the process of learning coding. These tools can be simple, like using different colored blocks to represent different types of commands in a program.	Building number sentences (G2) Coding for squirrel (G2) Coding the way (G4) Coding on a coordinate gride (G4) Let's paint our classroom (G4) Navigational coding (G5) Coding transformation (G6) Get the wolf to her pups (G7)
Oral	Introduction	The teacher outlines the objectives, explains the core concepts, and demonstrates how the activity relates to coding scenarios. For instance, the teacher might use storytelling, or real-life examples to illustrate complex coding principles in an understandable way.	Part2: mystery integer! (G6) Coding and composite number (G6)
	Discussion	The teacher might use open-ended questions to prompt students to think about the concepts in different ways.	Hen runs away from fox (G1) Scratch your probabilities (G4) Start small: making informed; financial decisions (G6) $20 or 20% (G7) Draw a maze (G7)

6.1.1.1 Unplugged is Kinesthetic. Kinesthetic refers to a learning approach centered on physical activities and movements (Lengel & Kuczala, 2010). Within kinesthetics, there are three sub-themes: body, dance, and tools.

The first sub-theme, *body*, focuses on using bodily movements and gestures to facilitate the learning of coding concepts (Lengel & Kuczala, 2010). This use of bodily movement to facilitate learning can be seen in the lesson plan "Please direct me to my seat" (Grade 1; Figure 13), where students are required to use their bodies to represent the "codes." Specifically, two students are first provided with a 5 x 5 grid on a floor with some obstacles (e.g., game tickets, drinks, hot dog, and the seats). Then one student (A) gives oral instructions (i.e., codes) to help another student (B) navigate around the obstacles and get to their seats. In this context, student B is using the body to enact the codes and understand what codes mean. Moreover, the approach (i.e., using bodies) may help students feel more engaged in the learning process (Lengel & Kuczala, 2010).

Figure 13

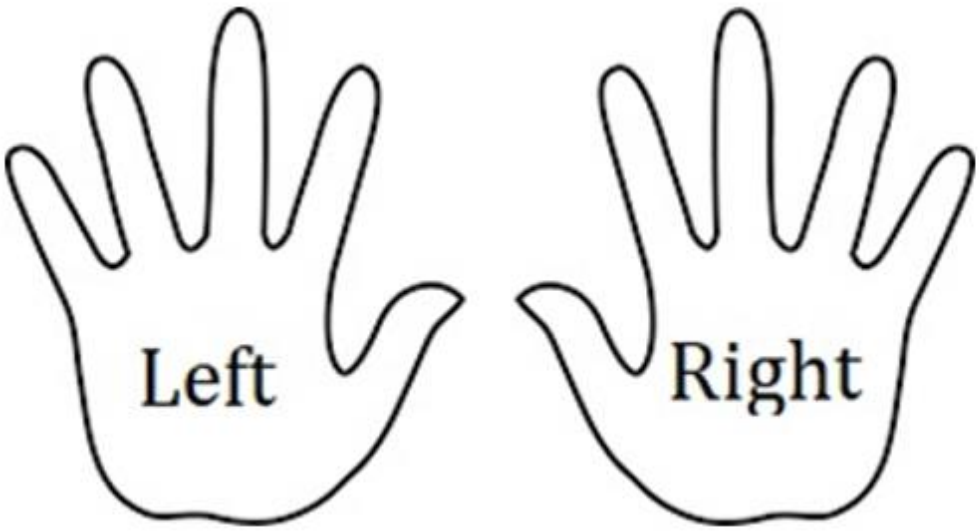
An Example from the Lesson Plan "Please direct me to my seat" (Grade 1)



Another example is the lesson plan "Barnyard commotion and motion," (Grade 1; Figure 14) which also involves physical actions (i.e., kinesiology) and collaborations (working with peers or teachers). In the lesson plan, students are first presented with a lot of instructions (e.g., rules, roles, figures, etc.). Then, the students are required to come up with a solution practiced by physical movements (e.g., hands, teams). They also need to work together with their peers to solve the problems. In this learning opportunity, multiple coding concepts may arise from the practice, such as pseudocoding and sequential events, as illustrated by this lesson plan.

Figure 14

An Example from Lesson Plan "Barnyard commotion and motion" (Grade 1)



The second sub-theme of kinesthetics, *dance,* is more than body movements; it is a narrative medium through which coding concepts can be articulated and understood by "bodies" (Hanna, 2008). In the lesson plan "Dancing to code" (Grade 2), dancing movements are emphasized (Table 9). This lesson plan encourages students to "create and perform a set of dances in a chosen sequence containing precise instructions for performing the dances" (Grade 2- Coding Lesson - Dancing to code, 2024, p. 2). Specifically, during the activity, the teachers' instructions serve as the "codes," requiring students to illustrate "sequences" through their dance moves. This method effectively engages students in the learning process (Flesch et al., 2021).

Table 9

An Example from the Lesson Plan "Dancing to code" (Grade 2)

Active Learning Part	
Teachers' instructions	Teacher moves
"Invite the students to regroup again Have them choose cards and create a dance set with the dances they selected" (Grade 2 - Coding Lesson - Dancing to code, 2024, p. 5).	"The teacher circulates and asks teams questions to get them arrange the cards for their sequence in an organized manner" (Grade 2 - Coding Lesson - Dancing to code, 2024, p. 5).

The final sub- theme, *tools,* refers to various unplugged resources used to facilitate hands-on interaction with coding concepts (e.g., Saxena et al., 2020). These tools range from simple items, such as pencils, pens, and paper, to more structured materials like grids or blocks. In the lesson plan, "Building number sentence" (Grade 2), the students are required to use physical materials like a 100's chart, counters, and sticky notes to create and understand number sentence. Figure 15 presents the details of the activity during the portion where the students can use the unplugged sticky notes to demonstrate sequences. Specifically, if they start at 50 on the chart and move back one square, they will reach 49. Alternatively, if they place sticky notes at 50 and 49, they will notice that there is a 1 square difference between these 2 numbers. Overall, the tools in this lesson plan are designed to simplify the understanding of sequence for students and lay the foundation for using "plugged devices" in subsequent activities (Grade 2 – Coding Lesson - Building number sentence, 2024, p. 3).

Figure 15

An Example from the Lesson Plan "Building number sentence" (Grade 2)

- Have students gather around the perimeter of the 100 number carpet or the 10x10 grid.
- Have a student pick a number. The teacher can use guiding sentences such as, "Pick a two digit even or odd number."
- Once a number is chosen (e.g., 49) place a counter on the number 49 to indicate that it is the chosen number.
- As a large group have a conversation around how we as a class can create a number sentence that equals 49 (e.g., 50 -1 = 49)
- Have the students use the unplugged sticky notes to create a sequential coding to show that if you start at 50 and go back 1 square you will get to 49. OR if you put a sticky note at 50 and 49 there is a 1 square difference between these 2 numbers.
- Now explain to the class that you are going to use the coding device to code to make a number sentence to represent 49 using an addition or subtraction strategy.
- Co-create a list of number sentences to make 49 and have the class take turns modelling the coding using the plugged coding device.

(30 minutes)

In summary, the theme kinesthetic refers to unplugged opportunities in which students can use bodies, dance, and tools to understand coding concepts. This approach may not only enhance students' comprehension and engagement in coding learning but also bridge the conceptual gap between theoretical knowledge and practical knowledge. It might be an effective strategy to introduce coding concepts in an accessible and engaging manner. Next, I will present the details of the second theme: unplugged is oral.

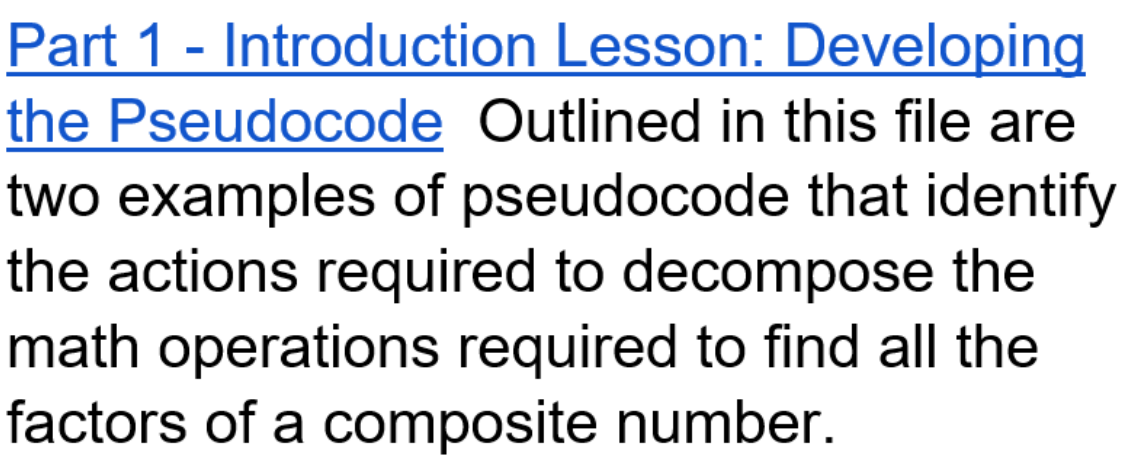
6.1.1.2 Unplugged is Oral. The second theme for unplugged activities, *oral*, is divided into two sub-themes: introduction and discussion. These two sub-themes emphasize the importance of verbal engagements and contribution to teaching and learning coding.

The first sub-theme, *introduction*, is the initial phase where the teachers introduce the coding concepts verbally. It aims to ensure all the learners have a solid understanding of the definitions of different coding concepts. For example, in the lesson plan "Coding and composite number" (Grade 6), the students start by conceptualizing their algorithm in pseudocode before coding in Scratch. Specifically, Figure 16 presents an instruction from the lesson where the

teacher is suggested to introduce two examples of pseudocode and the students will learn to express their math thoughts when decomposing an algorithm.

Figure 16

An Example from the Lesson Plan "Coding and composite number" (Grade 6)



The other sub-theme, *discussion*, always builds upon introduction. This sub-theme is represented by asking questions, answering questions, sharing ideas, and even debating various topics on coding. In the "$20 or 20%" lesson plan (Grade 7), the class is designed to begin with a discussion. Initially, the teacher presents a large poster (Figure 17). Subsequently, the teacher poses several questions to the students, including, "What are they telling you?", "What are they noy telling you?", and "What do you see?" (Grade 7 – Coding Lesson – $20 or 20%, 2024, p. 3). Then the students are required to discuss these questions with their peers. The lesson claims that this discussion will lead to the lesson's objective: creating a calculator that assists in choosing the more beneficial discount (Grade 7 – Coding Lesson – $20 or 20%, 2024, p. 4). In addition, the activity involves the coding concept of condition. Specifically, during the discussion of the

poster, students will be encouraged to apply this concept of condition to choose the more beneficial discount. In this context, the condition may serve as a decision-making criterion. This approach has the potential to teach students not only the technical aspect of coding— such as how to implement conditions in a program—but also critical thinking and decision-making skills.

Figure 17

An Example from the Lesson Plan "$20 or 20%" (Grade 7)

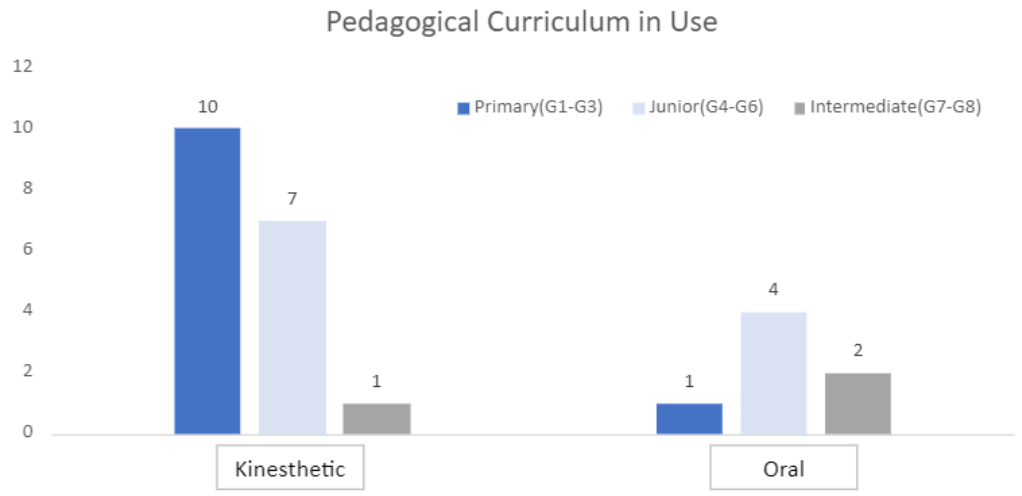
In conclusion, the theme of unplugged is oral includes aspects of the lessons that aim to help students understand the coding concepts before delving into actual coding. The lesson plans demonstrate that by actively engaging in dialogues, debates, and collaborative problem solving with their peers, students can solidify their understanding of coding concepts and principles. In

the next part, I will summarize the themes related to unplugged activities that were explored above.

6.1.1.3 Summary for Unplugged Curriculum in Use. In this section, I explored how students can learn coding through unplugged opportunities. Figure 18 presents the unplugged opportunities and their alignment with student development.

Figure 18

Bar Charts for Unplugged Curriculum in Use



Note. On the vertical axis, the two themes: kinesthetic and oral, emerge from the unplugged learning opportunities. The horizontal axis represents the number of lesson plans featuring these two themes. Specifically, when a lesson plan included both kinesthetic and oral activities, it is coded under the theme which is predominant. This approach ensures that each lesson plan is counted only once, based on its primary focus.

As the chart illustrates, kinesthetic forms are most used in lessons at the primary level (Grade 1 – Grade 3). By incorporating kinesthetic forms into the learning of coding, educators can leverage the tendencies of young learners to explore and understand not only the coding concepts but also the world around them (Strawhacker & Bers, 2019). This observation aligns

with Cheatum and Hammond's research (2000) which suggests that physical activities could effectively enhance students' learning experience. In addition, a systematic review research conducted by Sneck et al. (2019) supports the observation by stating that increased physical activities can positively affect young children's mathematics performance.

While kinesthetic forms are the mostly used for young learners (Grade 1- Grade 3), oral forms are mostly used in the lessons at the junior level (Grade 4 – Grade 6). This finding is echoed by Shiel et al.'s research (2012) which demonstrates that as children grow older, their cognitive and language abilities improve, allowing them to engage more deeply with oral forms of learning. Although lesson plan designers show a preference for using oral forms at the intermediate level (Grade 7 – Grade 8), the total number of unplugged opportunities (i.e., oral forms of learning) at that stage is smaller than that at the primary and junior levels. This trend suggests that by the time students reach the intermediate level, students are expected to have already acquired a foundational understanding of mathematics and coding; therefore, the need for unplugged coding opportunities (i.e., oral forms of learning) is reduced.

In conclusion, the application of unplugged opportunities in lesson plans is strategically aligned with the developmental stages of students, from kinesthetic forms in early stages to oral forms in later grades. These opportunities not only improve students' learning experiences through active engagement but also have the potential to support students' cognitive and language development, which demonstrates a positive impact on developing students' competencies. In the next part, I will explore the role of tinkering opportunities across all grades, which differ from unplugged coding opportunities by emphasizing the alteration of codes to observe different outcomes.

6.1.2 Tinkering for Curriculum in Use

As described in prior chapters, the goal of tinkering opportunities is to encourage experimentation, creativity, and learning through hands-on experience. Specifically, tinkering emphasizes a hands-on, exploratory opportunity to teach and learn coding within mathematics. I identified a total of 28 lesson plans applying tinkering opportunities to teach and learn coding. As shown in Table 10, two themes emerged from these tinkering opportunities: (1) Tinkering as code manipulation (n = 15); (2) Tinkering as interactive learning (n = 13).

Table 10

Tinkering for Curriculum in Use

Codes	Explanation	Lesson Plans
Code manipulation	It involves trying out different programming constructs, debugging, and refining the code to improve its functionality. And the tinkering process is not just about achieving specific goal, but also about exploring possibilities.	Coding with coins (G2) Dancing to code (G2) Building number sentences (G2) Coding a table of value (G3) Coding the way (G4) Scratch your probabilities (G4) Recalling math facts through coding (G5) Exploring patterns with fractions using Scratch (G5) Coding transformation (G6) Part1: which integer is greater (G6) Part2: mystery integer! (G6) Get the wolf to her pups (G7) Restaurant order app (G7) $20 or 20% (G7) Draw a maze (G7)
Interactive learning	In the process of tinkering as interactive learning, the learners interact with the material, and receive feedback from their actions. And the interaction is not just between the learner and the material or the tool but often involves interaction with peers (other learners). It is an	Story characters on a grid (G1) Please direct me to my seat (G1) Let's play hockey (G1) Barnard commotion and motion (G1)

	interactive process of testing ideas, observing results, and learning from successes and failures.	Hen runs away from fox (G1) Coding for squirrel (G2) Dance pattern (G3) Making change (G3) Dancing our way to coding using patterning (G4) Let's paint our classroom (G4) Budgets and spreadsheets: making plans (G5) Navigational coding (G5) Coding and composite number (G6)

6.1.2.1 Code Manipulation. The first theme, *code manipulation*, refers to the hands-on, experimental opportunities to understand the structure of the codes by directly manipulating it. This manipulation involves writing, adjusting, and understanding the codes to see how these changes affect the outcomes. In the lesson plan "Exploring patterns with fractions using Scratch" (Grade 5), students are given opportunities to modify Scratch codes to explore different fractions. Figure 19 displays an example where students are required to adjust parameters and sequences in the code (i.e., manipulate the code) and see real-time changes in how fractions are depicted. This hands-on approach not only solidifies their understanding of fractions but also encourages their problem-solving skills as they experiment with creating various patterns through coding.

Figure 19

An Example from the Lesson Plan "Exploring patterns with fractions using Scratch" (Grade 5)

While code manipulation allows learners to explore and play with codes, tinkering as interactive learning emphasizes the importance of creativity and interaction with contexts involved in the task. It will be explored in the next section.

6.1.2.2 Interactive Learning. The second theme, *interactive learning*, emphasizes engagement, interaction, and direct involvement with the materials being studied. This can be seen in the "Dance patterns" lesson plan (Grade 3), where students are required to interact with the codes and observe the outcomes. For example, the lesson provides students with three questions: "(1) What action was needed to start the circle pattern?", "(2) What action was needed to start the ballerina?", "(3) What would happen if we used the same action for both (either flag clicked, or space bar hit)? Alter the code and then?" (Grade 3 – Coding Lesson – Dance patterns, 2024, p. 7).

Tinkering activities in the lesson encourages students to experiment with code by changing actions (such as starting a circle pattern to initiate a ballerina's movement) and observing the outcomes. This tinkering process (i.e., experiment with code and observe the

outcomes) is not merely manipulating codes and is interactive and exploratory, thus allowing students to engage directly with materials (e.g., flag, space bar, etc.) and learn through trial and error.

Similar to "Dance patterns" (Grade 3), the "Let's paint our classroom" lesson plan (Grade 4) also requires students to think about different real-life situations by using "if, then". The two examples below illustrate the "if, then" logical structure by presenting real-word scenarios when making decisions based on specific conditions. Specifically, regarding the first one, the scenario shows the conditional logic in action: if I know the coverage of one can to paint, then I can calculate how many cans are necessary to paint the entire classroom by dividing the classroom's total area by the area one can covers. Regarding the second scenario, the conditional logic is used to calculate the expense: if the cost of one can is known, then the total cost can be calculated by multiplying this cost by the number of cans required.

> (1) "[If] one can of paint covers about 35 square meters of area, [then] how many cans are we going to need?" (Grade 4 – Coding Lesson – Let's paint our classroom, 2024, p. 5).
>
> (2) "[If] one can of paint cost around $50, [then] how much will it cost us to paint our classroom?" (Grade 4 – Coding Lesson – Let's paint our classroom, 2024, p. 5).

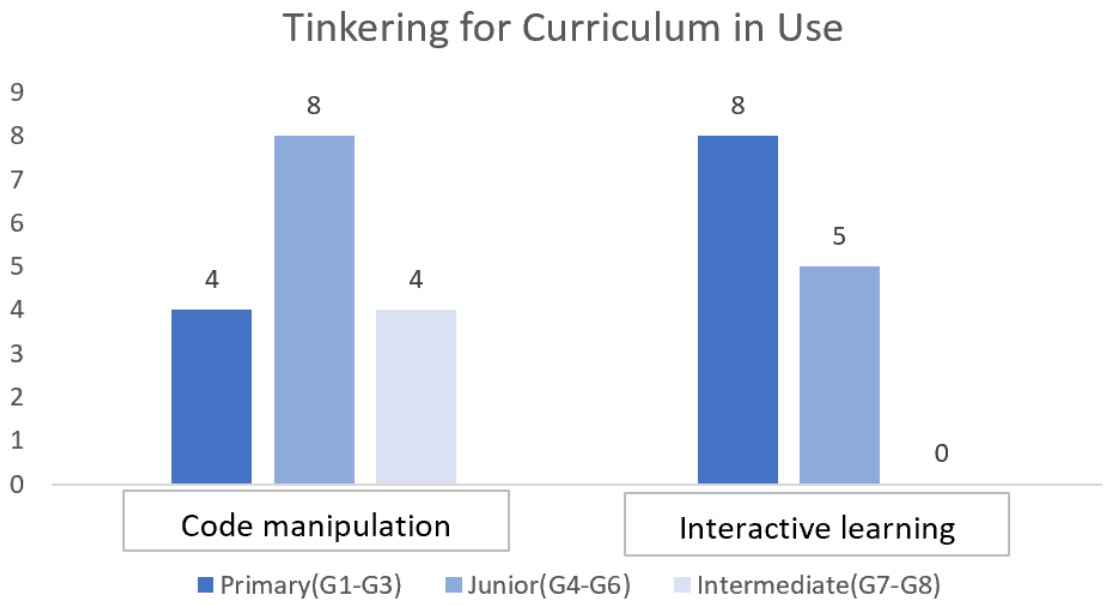
In this lesson plan, through tinkering with these activities, students have the potential to learn what the coding concept "if / then" is and also actively apply their knowledge to real-world situations, such as making a plan to paint the classroom. In this learning situation, the theme, interactive learning, emphasizes learning through doing, exploring, and making connections between knowledge and its application in real-world mathematics problem solving.

In conclusion, interactive learning activities emphasize the potential of integrating coding into real-life scenarios, allowing students to connect coding concepts to practical mathematics application. In the next section, I will summarize the tinkering opportunities that were explored above.

6.1.2.3 Summary for Tinkering Curriculum in Use. In this section, I explored how students can learn coding through tinkering. Figure 20 presents the tinkering opportunities and their alignment with student development.

Figure 20

Bar Charts for Tinkering Curriculum in Use



Note. On the vertical axis, the two themes: code manipulation and interactive learning, emerge from the tinkering learning opportunities. The horizontal axis represents the number of lesson plans featuring these two themes. Specifically, when a lesson plan included both code manipulation and interactive learning, it is coded under the theme which is predominant. This approach ensures that each lesson plan is counted only once, based on its primary focus.

Specifically, interactive learning is more prevalent in the lessons at the primary level (Grade 1 to Grade 3), with eight lesson plans applying tinkering as interactive learning. In contrast to the primary level, at the junior level (Grade 4 – Grade 6), the lesson plan designers apply tinkering as code manipulation and include seven lesson plans. At the intermediate level (Grade 7 – Grade 8), both code manipulation and interactive learning are in decline, with code manipulation used in four lesson plans and interactive learning not applied at all. In this way, the distribution of tinkering across different grades underscores a nuanced approach to teaching coding. While primary grades focus more on interactive learning to lay a foundational understanding of coding, the emphasis gradually shifts towards code manipulation in junior grades, reflecting an advanced engagement with coding concepts. However, there is a discernible decline in both practices by the intermediate level, suggesting space for other practices, which leads to the next section, "making coding learning opportunities." In this section, I explore how coding in the lessons encourages students to create new products to solve problems.

6.1.3 Making for Curriculum in Use

As described earlier, making refers to the process of creating something with a specific purpose in mind. When students are engaged in making, they aim to produce a final product or solution that meets predetermined requirements (Kotsopoulos et al., 2017). I identified a total of 29 lesson plans that create making opportunities to teach and learn coding. As shown in Table 11, this section focusses on two themes: (1) making as generalization (n = 10) and (2) making as digital making (n = 19).

Table 11

Making for Curriculum in Use

Codes	Sub-code	Explanation	Lesson Plans
Generalization	--	Generalization refers to the broad principles that can apply across different types of making.	Please direct me to my seat (G1) Barnard commotion and motion (G1) Dancing to code (G2) Building number sentences (G2) Dance patterns (G3) Coding the way (G4) Dancing our way to coding using patterning (G4) Budgets and spreadsheets: making plans (G5) Navigational coding (G5) Start small: making informed; financial decisions (G6)
Digital Making	Scratch	Scratch is a visual programming. It helps users learn the basics of coding logic without writing traditional codes.	Hen runs away from fox (G1) Making change (G3) Coding a table of value (G3) Coding on a coordinate grid (G4) Let's paint our classroom (G4) Scratch your probabilities (G4) Exploring patterns with fractions using Scratch (G5) Part1: which integer is greater (G6) Part2: mystery integer! (G6) Coding and composite number (G6) Draw a maze (G7) Coding and geometry (G8) Coding with Pythagorean theorem (G8)
	Google related tools	Digital making tools provided by Google for educational purposes, including Google Doc, Google forms, Google sheets, etc.	Recalling math facts through coding (G5)
	Others	Other tools, platforms, or methods that can be used, such as websites, games, etc.	Coding with coins (G2) Coding for squirrel (G2) Restaurant order app (G7) $20 or 20% (G7) Coding YouTube (G8)

6.1.3.1 Generalization. The first theme, *generalization*, refers to the process of synthesizing knowledge and experiences to grasp abstract concepts. Generalization can be observed in the lesson plan "Start small! Making informed financial decisions" (Grade 6). The activity requires students to revisit and rehearse the learning process, then generalize what they have learned, and finally consolidate their learning by creating a final project. Figure 21 presents the details of the consolidation step of generalization activity in this lesson plan.

Figure 21

An Example from the Lesson Plan "Start small! Making informed financial decisions" (Grade 6)

Consolidation of Learning

By using the information that students have brought forward through "**Starting Learning**" and "**Active Learning**" sessions, it's now time to connect their learning back to the learning goal and draw out some success criteria (anticipated and incidental).

Teacher prompt: Now that we have practised calculating generated funds and determining expenses associated with various ideas, let's think back to our Learning Goal. Our learning goal is 'I am learning to use and alter the existing budgeting codes in Google Sheets in order to help reach a financial goal.'

Together, the teacher and students can co-create the success criteria with the teacher which could then be turned into the assessment rubric.

Another example of generalization can be found in the lesson plan "Navigational coding" (Grade 5), where students work in pairs and create their own map of a town after learning related coding and mathematics concepts from Grade 1 to Grade 4. This activity serves as an exemplification of generalization, particularly within the aspect of "activity learning aspect" from the lesson plan (Grade 5 – Coding Lesson – Navigational coding, 2024, p. 4). By this stage,

students are expected to apply their knowledge acquired from previous grades (e.g., spatial reasoning, concurrent coding) to develop a tangible project like Figure 22 (Grade 5 – Coding Lesson – Navigational coding, 2024, p. 1–4)

Figure 22

An Example from the Lesson Plan "Navigational coding"(Grade 5)

In summary, the theme of generalization indicates parts of the lessons that encourage students to generalize their learning, connect it to real-world situations, and thus reinforce their coding and mathematics skills. These parts of the lessons involve students as active participants

in the learning process. Thus, by engaging students in generalizing, the making activities in the lessons encourage students to create a project based on acquired knowledge. Beyond generalization, I found that the making activities in the lessons also focus on a second theme of digital making by introducing new technologies to assist students in solving problems. I present results related to digital making in the next section.

6.1.3.2 Digital Making. The second theme, *digital making*, emphasizes the importance of technology in conducting coding activities (Kotsopoulos et al., 2017). I identified three sub-themes related to types of digital tools: Scratch, Google related tools, and others.

The first sub-theme, *Scratch*, is a visual programming language that targets children (https://scratch.mit.edu/). It allows learners to create interactive stories, games, and other projects using a simple block-based interface. These opportunities created with Scratch can be observed in the lesson plan "Exploring patterns with fractions using Scratch" (Grade 5), where students are tasked with using Scratch to gain a foundational understanding of fractions. Figure 23 provides the details for the task: With teachers' guidance, the students explore and apply patterns and observe how the size of the whole impacts the fraction size through coding in Scratch. This approach allows students to innovate and design their projects as well as develop their engagement and motivation towards learning coding.

Figure 23

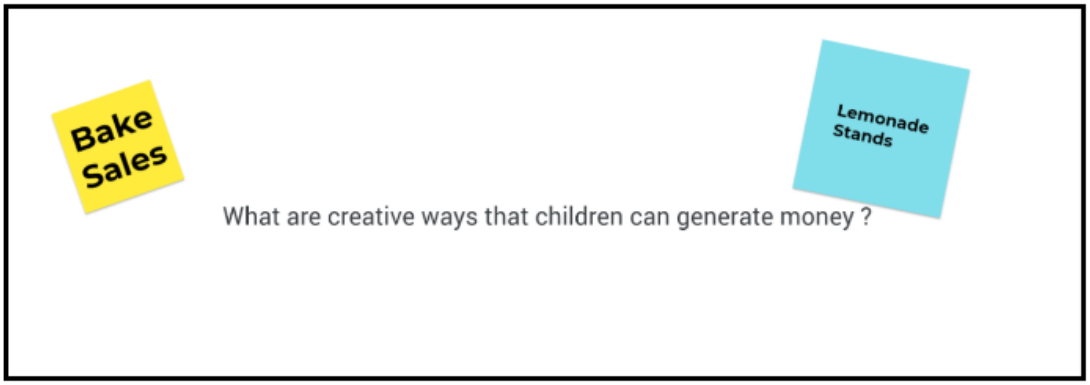
An Example from the Lesson Plan "Exploring patterns with fractions using Scratch" (Grade 5)

Note. This task includes instructions for teachers to ask, "What would happen if we set the denominator to 3 – so that we are adding 1/3 of the distance each time? Could you compare the result of this to the result with ½?" The lesson indicates that students may answer with, "The fraction (or decimal) getting added is smaller. It takes more steps (or terms of the patterns) to get close to 1 whole" or "It is similar because you still don't get to 1 whole when adding the fractions" (Exploring patterns with fractions using Scratch, 2024, p. 5).

The second sub-theme, *Google related tools*, is a free online toolkit, offering various tools, such as Jamboard (https://workspace.google.com/products/jamboard/). In the lesson plan "Start small: Making informed financial decisions" (Grade 6), students use Jamboard to create computational representations of financial situations. Figure 24 presents an example from the lesson where students are encouraged to engage in activities such as calculating earnings and expenses and assessing the viability of a financial plan using Jamboard. These activities require students to apply their knowledge of mathematics, financial literacy, and coding in a practical and digital format (Grade 6 – Coding Lesson – Start small! Making informed financial decisions, 2024, p. 1–2).

Figure 24

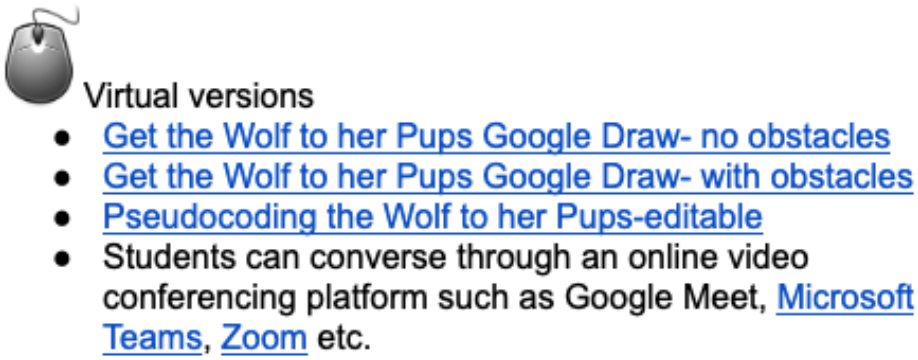
An Example from the Lesson Plan "Start small: making informed; financial decisions"(Grade 6)



In addition to the two popular tools mentioned above, digital making also includes a final sub-theme, *other tools*, such as communication tools, game designed tools, or calculation tools. The sub-theme of other tools is observed in the lesson plan "Get the wolf to her pups," (Grade 7) where the students are provided opportunities to collaborate with their peers through different platforms, such as Zoom, Microsoft, Google Meet (i.e., Google Meet is part of Google related tools, however, considering this lesson plan emphasizes the use of various collaborative platforms, such as Zoom, Microsoft, therefore, after discussing with my supervisor, we decided to categorize it under 'other tools'). Figure 25 presents details of the application of these platforms in the lesson. This example highlights how digital making can foster collaboration and allow students to work from different locations, share ideas in real-time, and collectively contribute to digital projects.

Figure 25

An Example from the Lesson Plan "Get the world to help her pups" (Grade 7)



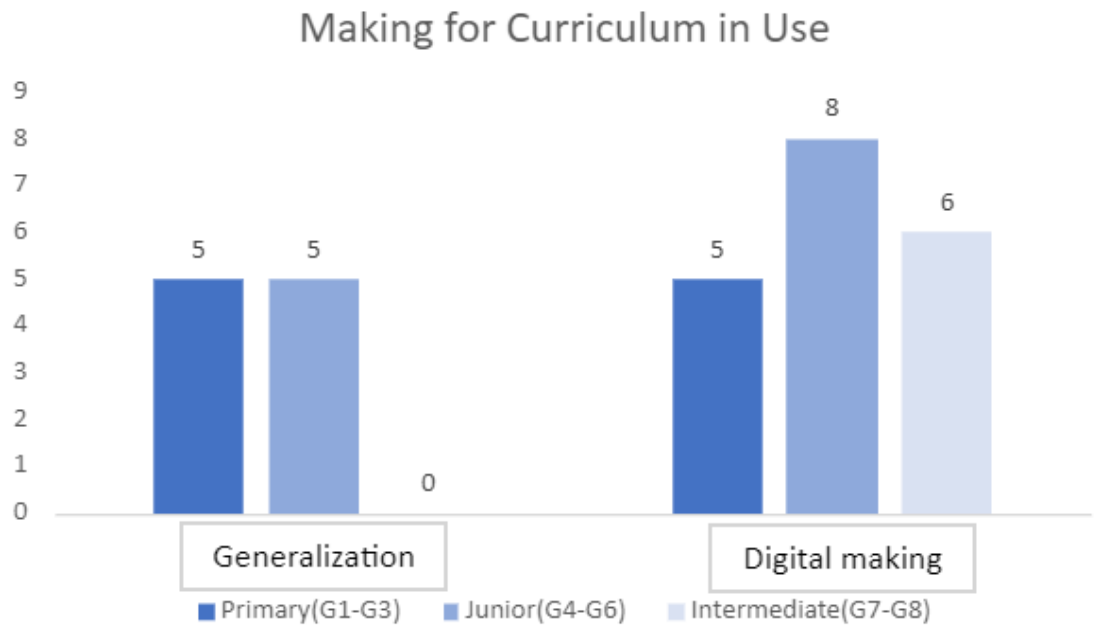
In summary, digital making activities in the lessons have the potential to enhance students' technology proficiency and encourages collaboration. Furthermore, these activities, as

presented in the lessons, aim to foster a learning environment where students can apply abstract coding concepts in a practical and innovative way, such as making a tangible project (e.g., making a town map) or applying various tools (e.g., Google related tools, Scratch). Next, I will present a summary of how students can learn coding through making.

6.1.3.3 Summary for Making Curriculum in Use. In this section, I summarize results of how students can learn coding through making within the lessons. Figure 26 presents the making opportunities and their alignment with student development.

Figure 26

Bar Charts for Making Curriculum in Use



Note. On the vertical axis, the two themes: generalization and digital making, emerge from the making learning opportunities. The horizontal axis represents the number of lesson plans featuring these two themes. Specifically, when a lesson plan included both generalization and digital making activities, it is coded under the theme which is predominant. This approach ensures that each lesson plan is counted only once, based on its primary focus.

As shown in the figure, I identified several trends in how making appears across grade levels in the lessons. Firstly, generalization is used equally at primary (Grade 1 – Grade 3) and junior levels (Grade 4 – Grade 6), but it is not used at the intermediate level (Grade 7 – Grade 8). Secondly, digital making is increasingly used as the educational level advances. Specifically, it is implemented in five lesson plans at the primary level (Grade 1 – Grade 3), eight at the junior level (Grade 4 – Grade 6), and six at the intermediate level (Grade 7 – Grade 8). The results indicate that as students progress through the grade levels, the focus of making activities shifts from generalization to digital making. The next section will explore how remixing can be used as coding learning opportunities, as presented in the lessons.

6.1.4 Remixing for Curriculum in Use

As described in previous chapters, remixing is a form of knowledge mobilization and requires students to identify usable parts of an object, then adapt and modify it to fit new purposes. As shown in Table 12, I identified two themes related to these remixing opportunities: (1) remixing is mobilizing knowledge (n = 6); (2) remixing is application in various contexts (n = 15).

Table 12

Remixing for Curriculum in Use

Codes	Description	Lesson plans
Mobilizing knowledge	It refers to the creative process of taking existing ideas, concepts, or materials to	Dance patterns (G3) Coding the way (G4)

	generate new knowledge (e.g., Levin, 2008).	Dancing our way to coding using patterning (G4) Coding on a coordinate gride (G4) Scratch your probabilities (G4) Part2: mystery integer! (G6) Coding YouTube (G8)
Application in various Contexts	It builds upon knowledge mobilization and encourages students to apply acquired knowledge to various contexts.	Lets' play hockey (G1) Banard commotion and motion (G1) Coding for squirrel (G2) Coding a table of value (G3) Budgets and spreadsheets: making plans (G5) Navigational coding (G5) Exploring patterns with fractions using scratch (G5) Start small: making informed; financial decisions (G6) Coding and composite number (G6) Restaurant order app (G7) $20 or 20% (G7) Draw a maze (G7) Coding with Pythagorean theorem (G8)

6.1.4.1 Mobilizing Knowledge. The first theme, *mobilizing knowledge***,** refers to the creative process of taking existing ideas, concepts, or materials to generate new knowledge (e.g., Levin, 2008). This can be observed in the lesson plan "Coding the way" (Grade 4) where students are required to use their understanding of coding principles to construct a new map that guides their partners. This activity embodies the theme "mobilizing knowledge" because through the act of designing a new map, students are not only applying what they have learned but also creating new things. Figure 27 presents the details of the activity within the lesson plan.

Figure 27

An Example from the Lesson Plan "Coding the way" (Grade 4)

Further Consolidation/Next Steps for students and teachers:

1. Students may wish to construct their own map on a blank grid and code the translations found within.

Blank grid virtual

2. Students may be challenged to work backwards from a set of coded translations in order to place landmarks/items on a grid.

3. Students may wish to guide a partner through a grid using a set of oral coded translations.

Similarly, in the lesson plan "Coding YouTube" (Grade 8), the students are required to create an infographic that displays their learning about monetizing a YouTube channel. The creating of an infographic is a form of knowledge mobilization that requires students to generalize the learning process and make it more accessible to various audiences.

Figure 28

An Example from the Lesson Plan "Coding YouTube" (Grade 8)

Further Consolidation/Next Steps for students and teachers:

Based on the consolidation information gathered from your students, teachers can further support learners by providing them with additional learning opportunities.
Such opportunities could include reviewing and reinforcing if/then statements through videos, tasks in a block based coding program (e.g., Scratch) where students could use existing tutorials/tasks to create games that involve if/then statements.

Infographic
Students can be encouraged to create an infographic displaying the learning they have had in regards to the monetizing of a YouTube channel. Infographic Tip Sheet . You can also look at the work of David McCandless and his work around the visualization of data. Link to TED Talk

In summary, the theme of mobilizing knowledge emphasizes the ways the lessons encourage students to reimagine and repurpose existing knowledge to create new insights, making them more accessible to broad audiences. In the following section, I will discuss the second theme, application in various contexts, which delves deeper into how this repurposed knowledge can specifically address the needs of various disciplines.

6.1.4.2 Application in Various Contexts. The second theme, *application in various contexts*, builds upon knowledge mobilization and encourages students to apply acquired knowledge to various contexts. This theme is observed in the lesson plan "Draw a maze" (Grade 7), where students are encouraged to apply their mathematics and coding knowledge acquired from the classroom in history and game design (Figure 29).

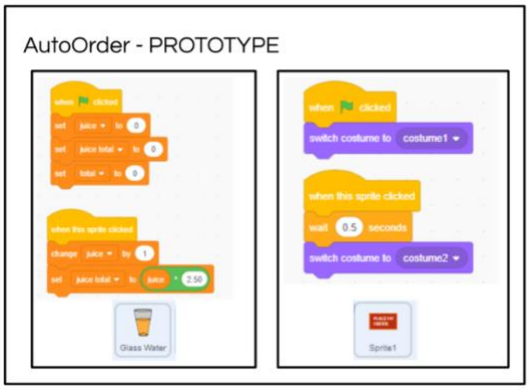
Figure 29

Maze of Potatoes/Labyrinthe de Pomme de Terre by OMAE-AFEMO (Grade 7)

Another example is the lesson plan "Restaurant order app" (Grade 7), where students are encouraged to apply their coding and mathematics knowledge gained in the lesson to design a restaurant game. Figure 30 presents this game design project that intersects with disciplines such as game design, UX (i.e., user experience) design, and marketing.

Figure 30

An Example from the Lesson Plan "Restaurant order app" (Grade 7)

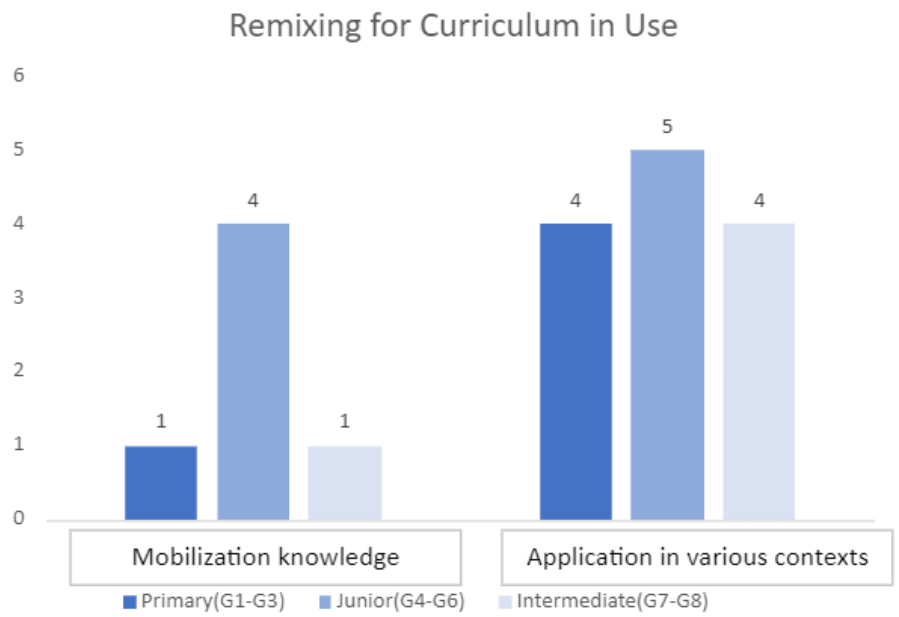


In sum, the theme of remixing as application in various disciplines emphasizes not only the transfer of knowledge from theoretical understanding to practical application but also encourages innovation and transdisciplinary collaboration. The inclusion of these types of activities has the potential to enhance students' understanding and application of coding in various disciplines (e.g., Lesson plan "Restaurant order app" for Grade 7 students, integrates mathematics, coding, and game design. Similarly, lesson plan "Draw a maze" for Grade 7 students, combines mathematics, coding, and history), thus preparing them for the complexities of real-world challenges. Next, I will present a summary of the remixing for coding learning opportunities.

6.1.4.3 Summary for Remixing Curriculum in Use. In this section, I explore how students can learn coding through remixing. Figure 31 represents the remixing opportunities and their alignment with student development.

Figure 31

Bar Charts for Remixing Curriculum in Use



Note. On the vertical axis, the two themes: mobilization knowledge and application in various contexts, emerge from the remixing learning opportunities. The horizontal axis represents the number of lesson plans featuring these two themes. Specifically, when a lesson plan included both mobilization knowledge and application in various contexts activities, it is coded under the theme which is predominant. This approach ensures that each lesson plan is counted only once, based on its primary focus.

Firstly, the theme, mobilization knowledge, is more emphasized at the junior level (Grades 4–6), while the theme, application in various contexts, is more evenly distributed across all the levels, with n = 4 at the primary level (Grades 1–3), n = 5 at the junior level (Grades 4–6), and n = 4 at the intermediate level (Grades 7–8). In conclusion, this observation suggests that the lesson plan designers applied remixing not only as a method to make coding more accessible, but also as a tool to encourage innovation, adaptability, and cross-disciplinary learning among students. In the next section, I will present a conclusion for this chapter.

6.2 Conclusion of Chapter 6

Chapter 6 explored how the Ontario math support lesson plans provided coding learning opportunities from four perspectives: unplugged, tinkering, making, and remixing. This exploration provided a detailed understanding of how coding learning opportunities are structured, extending from primary elementary to the more advanced stages in intermediate grades. In addition, it is important to note that these coding learning opportunities provided by the Ontario lesson plan designers are aligned with students' developmental stages, ensuring that these opportunities are both age-appropriate and challenging.

In the next chapter, I will discuss the results of Chapter 5 and Chapter 6 as well as the weaknesses and strengths of the conceptual framework used in this study.

Chapter 7: Discussion

This section discusses the results presented in Chapter 5 and Chapter 6. Specifically, I answer the two research questions of the study. The results of the first research question (How was coding conceptualized in the Ontario elementary mathematics curricula?) were presented in Chapter 5 while the results of the second research question (How was coding integrated into Ontario Math Support lesson plans?) were shown in Chapter 6.

7.1 Discussion on the First Research Question: How was Coding Conceptualized in the Ontario Elementary Mathematics Curricula?

Chapter 5 provided a comprehensive overview of the ways coding is conceptualized in the Ontario elementary mathematics curricula. Specifically, the Ontario mathematics curricula conceptualized coding as a multifaceted approach to teaching and learning mathematics as well as consisting of multiple coding concepts (Ministry of Education of Ontario, 2020; Ontario, 2022).

Firstly, coding is fundamental within Ontario's curricula (Ministry of Education of Ontario, 2020; Ontario, 2022). As illustrated in Chapter 5, coding is considered a key component in developing several skills for the 21st century, including mathematics thinking, problem solving, creativity, and digital literacy. It aims to cultivate a comprehensive skill set for students, applicable across various disciplines and real-life scenarios (Ministry of Education of Ontario, 2020). These aims of the curricula are aligned with the research conducted by Sheridan et al. (2016), which demonstrates that coding could help learners transfer their skills from school to workplace contexts.

Secondly, coding has been conceptualized as a progressive learning journey, evolving from simple, tangible activities to complex, real-world applications. This progressive approach emphasizes the importance of building a solid foundation in coding skills and concepts for students at young ages before advancing to more complex projects. It aligns with Piaget's (1953) perspectives, who argued that children develop intelligence by naturally progressing. A more recent study conducted by Strawhacker and Bers (2018) also supports this progressive curricular design. They state that integrating a coding and programming domain into developmental levels is a fundamental step for creating evidence-based computer science curricula (Strawhacker & Bers, 2018).

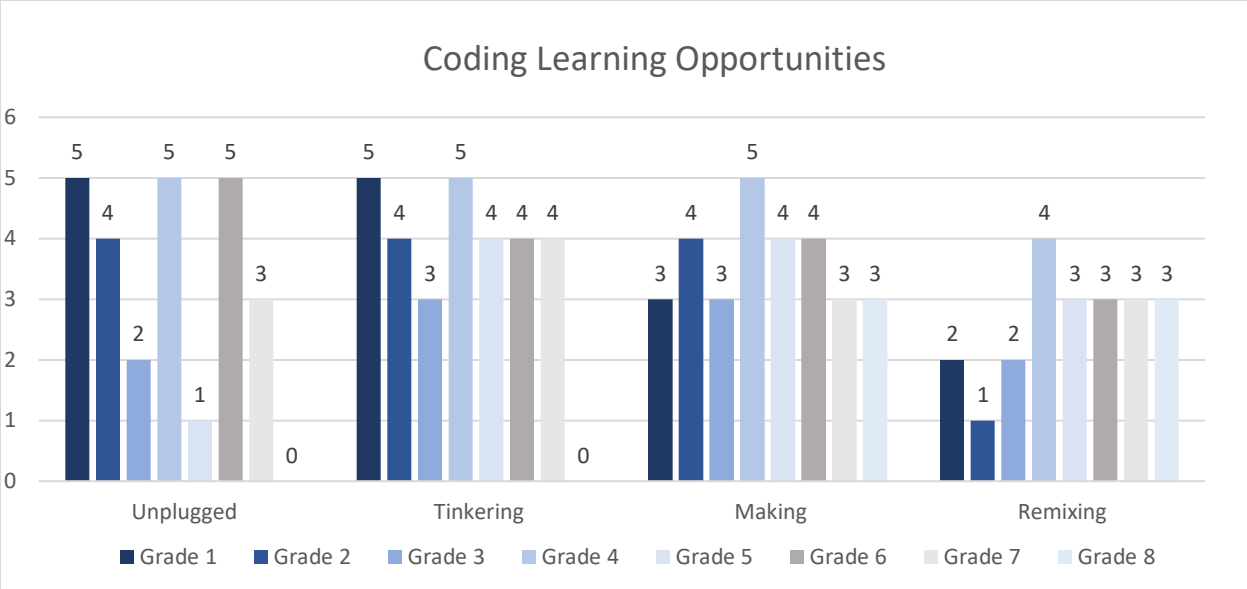
In summary, coding in the Ontario curricula is conceptualized as a comprehensive approach that goes beyond merely teaching technical or mathematics skills (Ontario, 2022). It aims to enhance students' mathematics learning and prepare them for a technology-rich society. Most importantly, coding is part of a developmental learning process that supports students' growth across various disciplines.

7.2 Discussion on the Second Research Question: How was Coding Integrated into Ontario Math Support's Lesson Plans?

In Chapter 6, four opportunities to learn coding were presented: unplugged, tinkering, making, and remixing. Figure 32 illustrates the progression of four opportunities across the grades.

Figure 32

Bar Charts for Coding Opportunities in Lesson Plans across Grade Levels



Note. My unit of the analysis is the lesson plan. That is, the numbers on the vertical axis represent the numbers of lesson plans that contained each specific learning opportunity. Additionally, each lesson plan is counted once per coding learning opportunity. For example, a lesson plan that comprises several unplugged activities is coded once under the unplugged learning opportunity category.

Notably, tinkering is the most consistently emphasized opportunity, maintaining a high frequency across all grade levels, except for Grade 8. In contrast, remixing exhibits its greatest prominence at Grade 1, decreases sharply from Grade 1 to Grade 2, then increases, maintaining a steady presence from Grade 4 onwards. Unplugged opportunities are notably prevalent in the lower grades, specifically from Grades 1 to 6, whereas making opportunities persist across all grades. Below, I explain why these trends may happen.

Unplugged opportunities are frequently used in the lower grades (Grades 1 to 6). One possible explanation is that unplugged opportunities often involve physical activities or visual representations, which are more suitable for students who may not yet have the skills to use digital devices. This reasoning is substantiated by Lee and Junoh's (2019) research, which suggests that it is important to purposefully and systematically help children become familiar with coding and digital literacy from a very young age. In addition, their study emphasizes the

importance of presenting coding in a developmentally appropriate way (Lee & Junoh, 2019). Thus, when integrating coding into lessons, particularly for students at young ages, it is vital for researchers and educators to ensure that the approach engages them playfully.

Tinkering and making opportunities are used more often in the lesson plans compared to unplugged and remixing opportunities. Specifically, tinkering encourages experimentation and iterative learning. Making supports students by applying digital creation in solving problems. The characteristics of these two opportunities align with the academic resilience (AR) theory, as AR encourages students to persist in the face of difficulties, adaptively overcome obstacles, and continuously improve their understanding and skills through repeated efforts (Wang et al., 1994). AR, therefore, underpins the educational value of tinkering and making by highlighting the importance of perseverance, adaptability, and the willingness to engage with challenging problems, thereby enhancing students' likelihood of academic success.

Remixing is the least used among the four opportunities in the Ontario lesson plans. This might be because remixing involves transferring knowledge and applying acquired knowledge from mathematics into different disciplines. This opportunity might be less familiar and more complex for both educators and students compared to those more straightforward opportunities (i.e., unplugged, tinkering, and making) (Kotsopoulos et al., 2017). In addition, the interdisciplinary nature of remixing might require both the educators and students to have a broad understanding and the ability to connect coding and mathematics skills from various fields (Dasgupta et al., 2016). Consequently, there is an urgent need for developing professional programs that equip educators with the skills to transfer coding concepts and skills from mathematics to other disciplines (Kong & Wong, 2017). Kong and Wong (2017) emphasize the importance of preparing teachers to introduce CT and coding into school subjects. Similarly, a

study conducted by Yadav et al (2017) also supports this recommendation. They suggest that preparing teachers for CT would empower them to prepare their students for the tech-rich society (Yadav et al., 2017).

In conclusion, these four opportunities offer a multifaced approach to teaching and learning coding within mathematics contexts. They enhance students' understanding of mathematics and coding concepts and equip students with essential skills in both disciplines. Moreover, these opportunities have the potential to create a learning environment that values experimentation, creativity, and collaboration, which aligns well with the demands of 21st-century education.

7.3 Strengths and Weakness of Conceptual Framework

As detailed in Chapter 3, I developed a conceptual framework (Figure 3) for conducting this research. Although the conceptual framework has a systematic structure for understanding how to analyze coding integration in the Ontario mathematics curricula, it may require additional elements and clarifications to be effectively operationalized for my future research. Below, I present the strengths and weaknesses of the conceptual framework.

Strengths: There are two significant strengths in using this conceptual framework. Firstly, it provides an understanding of coding integration from two perspectives. Specifically, the framework combines curriculum analysis and content knowledge for teaching, providing an understanding of Ontario curricula which extends from official curricula to curriculum in use. Through the combination, in my research, I gained insights into both the theoretical perspectives and practical applications of the Ontario curricula. Secondly, I used a pedagogical framework to

guide the analysis of coding learning opportunities. It offered a structured approach to identifying and understanding the diverse methods through which coding opportunities are created (by lesson plan designers) for the students. The framework also has potential to support my future research. By delineating different types of coding learning opportunities, this framework may provide me with a comprehensive toolkit for designing and implementing coding interventions and will also allow me to explore more themes and subthemes related to how to learn coding.

Weakness: I found that there were two weaknesses in using the conceptual framework. Firstly, the conceptual framework is not designed to encompass a diverse range of participants, including students, teachers, and educators. By focusing solely on the analysis of official documents and lesson plans, the study may overlook critical insights and feedback from those actively engaged in the educational process. Second, the framework did not include ways to critique of the curricula, which limits the study's ability to provide recommendations for effectively refining the Ontario curricula.

In conclusion, while the conceptual framework I developed effectively combines curriculum analysis, content knowledge for teaching, and a pedagogical framework to offer a comprehensive understanding and practical application for coding integration within mathematics contexts, it lacks inclusivity regarding diverse participants and a lens for thorough critique of curricula, which limits its operational effectiveness for curriculum refinement in future research. The next section will discuss the limitations of the master's thesis.

7.4 Limitations

This master's thesis study has some limitations. Firstly, these results could not be generalized into other settings (e.g., other provinces in Canada, other countries in the world). Specifically, the curricular documents and lessons in other provinces are different from Ontario. Consequently, the findings from this study may not be applicable to other provinces.

In addition, considering that the research relies on curriculum analysis, I did not interview teachers, teacher educators, and students within the Ontario educational systems. Therefore, although the results provide a valuable reference for researchers interested in the integration of coding and mathematics, this research does not examine the effectiveness of coding integration for mathematics in actual classroom settings.

Finally, the ways that I defined coding concepts may limit the generalizability of the results. Specifically, I have defined concepts for coding and coding integration at the beginning of research (Chapter 1 and Chapter 2). This approach limits this research's scope to explore diverse definitions and perspectives of coding within mathematics contexts. Therefore, the results cannot be applied to other educational settings that have different definitions of coding.

In conclusion, this master's thesis presents valuable insights into the conceptualization and integration of coding within the Ontario elementary mathematics curricula and some of their publicly available lessons. However, the study's results are subject to some limitations that must be recognized. The specificity of the Ontario context, with its unique educational frameworks, restricts the generalizability of the results to other regions or educational settings. This limitation underscores the importance of considering local contexts when interpreting the study's findings.

Chapter 8: Conclusion

In this final chapter, I first summarize the findings. Specifically, I revisit the entire study and connect its findings with existing literature, highlighting how my insights build upon and diverge from previous research. Furthermore, I articulate the specific contributions of my research to the field and outline directions for future investigations.

8.1 Summary of the Findings

This qualitative research investigated how coding has been conceptualized and integrated into the Ontario mathematics curricula and Ontario Math Support's lesson plans from Grades 1 to 8. Two major findings emerged from the analysis of the Ontario documents, specifically the official Ontario curricula and the Ontario Math Support's lesson plans. Each of the findings corresponds to one of the two research questions discussed in Chapter 7, sections 7.1 and 7.2.

The first finding presented in Chapter 7, section 7.1, indicates that coding skills are emphasized in the Ontario curricula and are developed sequentially within the curricula documents. The results that support this finding were presented in Chapter 5 for each of the three cases: Case 1 (primary elementary; Grades 1 to 3); Case 2 (junior elementary; Grades 4 to 6); and Case 3 (intermediate elementary; Grades 7 to 8). Specifically, in Case 1, the youngest students must focus on foundational coding concepts that employ unplugged methods or visual languages. This focus has the potential to lay a foundation for understanding more complex coding concepts in the later grades. Following this, in Case 2, students are expected to build upon primary elementary levels and apply coding to enhance mathematics skills. Moreover, this phase is a transition between primary elementary and intermediate elementary. Then, in Case 3,

students are expected to apply coding into various disciplines. This stage aims to solidify students' transdisciplinary skills, such as digital literacy and data analytical skills, thus potentially preparing them for the 21st century. This developmental sequence reflects existing theories in developmental psychology, especially those concerning students' cognitive development (Strawhacker & Ber, 2019; Marinus et al., 2018; Gluga et al., 2013; Hwang et al., 2008). These theories suggest that students benefit most from learning experiences that are tailored to their current level of understanding, then gradually increase in complexity as their skills develop (Piaget, 1953). In sum, this finding emphasizes the potential of coding curricula to adapt to the developmental stages of learners, potentially providing a solid foundation for lifelong learning and adaptability in a rapidly changing digital world.

My second finding presented in Chapter 7, section 7.2, demonstrates that the Ontario Math Support's math lessons provide diverse opportunities to integrate coding into the teaching and learning process. The results that support this finding were presented in Chapter 6 where four opportunities were identified: unplugged, tinkering, making, and remixing, each offering unique benefits. The ways that these learning opportunities were integrated across grade levels reflected recommendations and practices described in existing literature in CT (Bers, 2018; Bers, 2018). Specifically, in the lessons, unplugged is incorporated as an opportunity for young students to learn coding concepts, which is aligned with Brackmann et al.'s (2017) research where unplugged can effectively enhance primary students' CT skills. In addition, tinkering (i.e., manipulating codes), which appeared across most of the grade levels, is similar to debugging (i.e., fixing errors to achieve the best performance of codes) (e.g., Kong & Wang, 2023) – a core practice of CT and a vital skill for computer programmers to grasp (Kong & Wang, 2023; Shute et al., 2017; Brennan & Renisck, 2012). Furthermore, this study identified making as a creative

method for introducing new technologies. Additionally, this study indicated that it had the potential to promote collaborative learning. Herro et al.'s (2021) research supported this argument, stating that making is an efficient way to develop social and CT skills, owing to its emphasis on collaborative problem solving. Finally, I identified that remixing was used in the Ontario lessons as an approach for students to apply their acquired knowledge from mathematics to various disciplines – something that can lead to CT learning. This potential for remixing to support CT is supported by Dasgupta et al.'s (2016) study. They propose that remixing, involved various domains, such as music, videos, and other interactive tools, and can be regarded as a pathway to learning CT (Dasgupta et al. 2016). In sum, the second finding shows that the Ontario lessons align with the broader goals of CT education that aim to equip students to master CT concepts, CT practices, and CT perspectives across various domains (Brennan & Renisck, 2012), potentially preparing them for a rapidly evolving society.

In summary, these two findings collectively enrich the existing literature on curriculum studies and CT. Most importantly, these findings demonstrate how teachers and educators can integrate coding into practices, potentially preparing students for the rapidly evolving world. In the next and last section, I will present the implications and directions for future research.

8.2 Implications & Future Research

In conclusion, this research has practical and theoretical implications. Practically, it provides valuable guidance/reference for multiple communities, including in-service and pre-service teachers, school administrators, policymakers, and educational researchers on how to integrate coding into teaching and learning mathematics. The proposed curriculum development identified in the study, advocates for a progressive, deliberate inclusion of coding skills and

concepts, aligned with cognitive development theories (Piaget, 1953). It has the potential to help students progressively build their coding and mathematics competencies in harmony with their developmental stages. Moreover, by incorporating a diverse range of learning opportunities, including unplugged, tinkering, making, and remixing, the curriculum in use has the potential to cater to various learning styles and preferences, thus providing a rich and varied learning experience for students across grades. For instance, kinesthetic learning opportunities (a form of unplugged learning opportunity), frequently used at lower grades, could potentially enhance students' engagement and collaboration in learning coding and mathematics. Therefore, the researchers intending to develop young kids' collaborative and coding skills could consider creating interventions that focus on kinesthetic formats, such as dancing, tools, and body movements.

Theoretically, I have expanded the CT model developed by Kotsopoulos et al. (2017) by identifying related themes and subthemes. This potentially broadens its scope and applicability within the context of elementary mathematics. Specifically, the expansion categorizes coding learning opportunities into four main areas: unplugged, tinkering, making, and remixing. Each category is supported by distinct thematic elements, further refined into subthemes when applicable. Unplugged opportunities emphasize kinesthetic engagement (with Body, Dance, and Tools subthemes) and Oral interactions (with Introduction, and Discussion subthemes). Tinkering involves hands-on Code manipulation and Interactive learning, both without further subthemes. Making is divided into Generalization and Digital making (with subthemes Scratch, Google-related tools, and Other tools). Finally, Remixing includes Mobilizing knowledge and Application in various contexts, both without subthemes. This framework not only deepens the

coding integration into elementary mathematics curricula but also proposes a structured approach to cultivating versatile skill sets for elementary students.

Regarding future research, I plan to use the CCKMTL framework developed in the study to explore how curriculum policies can be designed to effectively integrate coding into various educational programs (e.g., mathematics, science, special education). My approach will involve reviewing policy documents from different countries and interviewing policymakers and educators worldwide. In addition, based on the CCKMTL framework, I aim to compare various international models of education that have successfully integrated coding. Specifically, I am interested in conducting a comparative analysis between China (i.e., Eastern culture) and Canada (i.e., Western culture) to understand how cultural differences impact the efficacy of coding integration in education.

References

Albanese, M. A., Mejicano, G., Mullan, P., Kokotailo, P., & Gruppen, L. (2008). Defining characteristics of educational competencies. *Medical Education*, *42*(3), 248-255. https://doi.org/10.1111/j.1365-2923.2007.02996.x

Alberta Education. (2017). DRAFT kindergarten to grade 12 mathematics scope and sequence. Retrieved from. http://www.abcee.org/sites/abcee.org/files/Scope%20and%20Sequence%20Math.pdf

Albion, P. R. (2016). The second coming of coding: Will it bring rapture or rejection? S. Prestidge, P. Albion (Eds.), In *Australian Council for Computers in Education 2016 Conference Refereed Proceedings*. ACCE, Brisbane (2016), pp. 1-8. Retrieved from: http://conference.acce.edu.au/index.php/acce/acce2016/paper/view/49

Ball, D. L., Thames, M. H., & Phelps, G. (2008). Content knowledge for teaching: What makes it special?. *Journal of Teacher Educatio*n, *59*(5), 389-407. https://doi.org/10.1177/0022487108324554

Barr, D., Harrison, J., & Conery, L. (2011). Computational thinking: A digital age skill for everyone. *Learning & Leading with Technology*, *38*(6), 20-23.

Battal, A., Afacan Adanır, G., & Gülbahar, Y. (2021). Computer science unplugged: A systematic literature review. *Journal of Educational Technology Systems*, *50*(1), 24-47. https://doi.org/10.1177/0047239521101880

Bernardo, M. A., & Morris, J. D. (1994). Transfer effects of a high school computer programming course on mathematical modeling, procedural comprehension, and verbal

problem solution. *Journal of Research on Computing in Education*, *26*(4), 523-536. https://doi.org/10.1080/08886504.1994.10782108

Bers, M. U. (2018). Coding, playgrounds and literacy in early childhood education: The development of KIBO robotics and ScratchJr. *2018 IEEE global engineering education conference (EDUCON), Spain,* 2094-2102. https://doi.org/10.1109/EDUCON.2018.8363498

Bers, M. U. (2018). Coding and computational thinking in early childhood: The impact of ScratchJr in Europe. *European Journal of STEM Education*, *3*(3), 08:1-13.

Boaler, J. (1993). The role of contexts in the mathematics classroom: Do they make mathematics more "real"?. *For the Learning of Mathematics*, *13*(2), 12-17.

Boyabatlı, O., Leng, T., & Toktay, L. B. (2016). The impact of budget constraints on flexible vs. dedicated technology choice. *Management Science*, *62*(1), 225-244. https://doi.org/10.1287/mnsc.2014.2093

Brackmann, C. P., Román-González, M., Robles, G., Moreno-León, J., Casali, A., & Barone, D. (2017). Development of computational thinking skills through unplugged activities in primary school. In E. Barendsen, & P. Hubwieser (Eds.), WiPSCE' 17: *Proceedings of the 12th Workshop on Primary and Secondary Computing Education* (pp. 65–72). ACM. https://doi.org/10.1145/3137065.3137069

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, Vol. 2. Research designs: Quantitative, qualitative, neuropsychological, and*

biological (pp. 57–71). American Psychological Association. https://doi.org/10.1037/13620-004

British Columbia Ministry of Education. (2016). Applied design, skills, and technologies [Program of Studies]. Retrieved from https://curriculum.gov.bc.ca/curriculum/adst

Brennan, K., & Resnick, M. (2012, April 13–17). *New frameworks for studying and assessing the development of computational thinking* [Conference presentation]. AERA 2012 Conference, Vancouver, BC, Canada.

Browning, C., Edson, A. J., Kimani, P., & Aslan-Tutak, F. (2014). Mathematical content knowledge for teaching elementary mathematics: A focus on geometry and measurement. *The Mathematics Enthusiast*, *11*(2), 333-383. https://doi.org/10.54870/1551-3440.1306

Burgmann (2016). *B.C. to add coding to K-12 school curriculum. CTV NEWS.* Retrieved from https://www.ctvnews.ca/politics/b-c-adds-coding-to-k-12-school-curriculum-1.2742593

Calder, N., & Rhodes, K. (2021). Coding and learning mathematics: How did collaboration help the thinking?. In Y. H. Leong, B. Kaur, B. H. Choy, J. B. W. Yeo, & S. L Chin (Eds.), *Excellence in Mathematics Education: Foundations and Pathways (Proceedings of the 43rd annual conference of the Mathematics Education Research Group of Australasia)* (pp. 139-146). Singapore: MERGA.

Ceylan, M., & Aslan, D. (2023). The effect of learning trajectories-based coding education program on preschoolers' mathematical measurement skills. *Education and Information Technologies*, 1-21. https://doi.org/10.1007/s10639-023-12107-7

Chan, S. W., Looi, C. K., Ho, W. K., & Kim, M. S. (2023). Tools and approaches for integrating computational thinking and mathematics: A scoping review of current empirical studies. *Journal of Educational Computing Research*, *60*(8), 2036-2080. https://doi.org/10.1177/07356331221098793

Cheatum, B. A., & Hammond, A. A. (2000). *Physical activities for improving children's learning and behavior: A guide to sensory motor development*. Human Kinetics. https://scholarworks.wmich.edu/books/525

Connelly, F. M., & Connelly, G. (2012). Curriculum policy guidelines: Context, structures and functions. In A. Luke., A. Woods., K. Weir (Eds.), *Curriculum, syllabus design and equity: A primer and model* (1st ed., pp. 64-83). Routledge. https://doi.org/10.4324/9780203833452

Dai, C. P., Ke, F., Pan, Y., & Liu, Y. (2023). Exploring students' learning support use in digital game-based math learning: A mixed-methods approach using machine learning and multi-cases study. *Computers & Education*, *194*, Article 104698. https://doi.org/10.1016/j.compedu.2022.104698

Dasgupta, S., Hale, W., Monroy-Hernández, A., & Hill, B. M. (2016). Remixing as a pathway to computational thinking. *Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing, USA,* 1438-1449. https://doi.org/10.1145/2818048.2819984

Demirer, V., & Sak, N. (2016). Programming education and new approaches around the world and in Turkey. *Eğitimde Kuram ve Uygulama. 12*(3), 521-546.

Dohn, N. B. (2020). Students' interest in Scratch coding in lower secondary mathematics. *British Journal of Educational Technology*, *51*(1), 71-83. https://doi.org/10.1111/bjet.12759

Dralle-Moreano, J. (2021). *Educators' perceptions of benefits and barriers of the inclusion of coding in K-8 curriculum: A qualitative study* [Doctoral dissertation, St. John's University]. https://scholar.stjohns.edu/theses_dissertations/170

Drisko, J. W., & Maschi, T. (2016). *Content Analysis*. Oxford University Press.

Edwards, R., Biesta, G., & Thorpe, M. (Eds.). (2009). *Rethinking Contexts for Learning and Teaching: Communities, Activities and Networks* (1st ed.). Routledge. https://doi.org/10.4324/9780203881750

Egbert, J., Shahrokni, S. A., Abobaker, R., & Borysenko, N. (2021). "It's a chance to make mistakes": Processes and outcomes of coding in 2nd grade classrooms. *Computers & Education*, *168*, Article 104173. https://doi.org/10.1016/j.compedu.2021.104173

Flesch, B., Gabaldón, C., Nabity, M., & Thomas, D. (2021). Choreographing increased understanding and positive attitudes towards coding by integrating dance. *International Journal of Computer Science Education in Schools*, *4*(3), 31-48. https://doi.org/10.21585/ijcses.v4i3.109

Fossey, E., Harvey, C., McDermott, F., & Davidson, L. (2002). Understanding and evaluating qualitative research. *Australian & New Zealand Journal of Psychiatry*, *36*(6), 717-732. https://doi.org/10.1046/j.1440-1614.2002.01100.x

Fraillon, J., Ainley, J., Schulz, W., Friedman, T., & Duckworth, D. (2020). *Preparing for life in a digital world: IEA international computer and information literacy study 2018 international report*. Springer.

Francis, K., Bruce, C., Davis, B., Drefs, M., Hallowell, D., Hawes, Z., McGarvey, L., Moss, J., Mulligan, J., Okamoto, Y., Sinclair, N., Whiteley, W., & Woolcott, G. (2017). Multidisciplinary perspectives on a video case of children designing and coding for robotics. *Canadian Journal of Science, Mathematics and Technology Education*, *17*, 165-178. https://doi.org/10.1080/14926156.2017.1297510

Gadanidis, G., Cendros, R., Floyd, L., & Namukasa, I. (2017). Computational thinking in mathematics teacher education. *Contemporary Issues in Technology and Teacher Education, 17*(4), 458-477.

Gim, N. G. (2021). Development of life skills program for primary school students: Focus on entry programming. *Computers*, *10*(5), 56. https://doi.org/10.3390/computers10050056

Gluga, R., Kay, J., Lister, R., Simon, & Kleitman, S. (2013). Mastering cognitive development theory in computer science education. *Computer Science Education, 23*(1), 24-57. https://doi.org/10.1080/08993408.2013.768830

Government of Canada. (2022, January). *Education in Canada: Types of Schooling*. Government of Canada. https://www.canada.ca/en/immigration-refugees-citizenship/services/new-immigrants/new-life-canada/education/types-school.html

Govind, M., Relkin, E., & Bers, M. U. (2020). Engaging children and parents to code together using the ScratchJr app. *Visitor Studies*, *23*(1), 46-65. https://doi.org/10.1080/10645578.2020.1732184

Grade 1 – Coding Lesson – Storybook characters on a grid [Sample lesson plans]. (n.d.). Math Curriculum Resource Project. Retrieved March 1, 2024, from https://ontariomath.support/?pg=results&type=subject&lang=EN&subject=Coding

Grade 1 – Coding Lesson – Please direct me to my seat [Sample lesson plans]. (n.d.). Math Curriculum Resource Project. Retrieved March 1, 2024, from https://ontariomath.support/?pg=results&type=subject&lang=EN&subject=Coding

Grade 1 – Coding Lesson – Les's play hockey [Sample lesson plans]. (n.d.). Math Curriculum Resource Project. Retrieved March 1, 2024, from https://ontariomath.support/?pg=results&type=subject&lang=EN&subject=Coding

Grade 1 – Coding Lesson – Barnyard Commotion and motion [Sample lesson plans]. (n.d.). Math Curriculum Resource Project. Retrieved March 1, 2024, from https://ontariomath.support/?pg=results&type=subject&lang=EN&subject=Coding

Grade 1 – Coding Lesson – Hen Runs Away from Fox [Sample lesson plans]. (n.d.). Math Curriculum Resource Project. Retrieved March 1, 2024, from https://ontariomath.support/?pg=results&type=subject&lang=EN&subject=Coding

Grade 2 – Coding Lesson – Coding with coins [Sample lesson plans]. (n.d.). Math Curriculum Resource Project. Retrieved March 1, 2024, from https://ontariomath.support/?pg=results&type=subject&lang=EN&subject=Coding

Grade 2 – Coding Lesson – Dancing to code/Coding to dance [Sample lesson plans]. (n.d.). Math Curriculum Resource Project. Retrieved March 1, 2024, from https://ontariomath.support/?pg=results&type=subject&lang=EN&subject=Coding

Grade 2 – Coding Lesson – Building number sentences [Sample lesson plans]. (n.d.). Math Curriculum Resource Project. Retrieved March 1, 2024, from https://ontariomath.support/?pg=results&type=subject&lang=EN&subject=Coding

Grade 2 – Coding Lesson – Coding for the squirrel [Sample lesson plans]. (n.d.). Math Curriculum Resource Project. Retrieved March 1, 2024, from https://ontariomath.support/?pg=results&type=subject&lang=EN&subject=Coding

Grade 2 – Coding Lesson – Location and movement-Coding ourselves [Sample lesson plans]. (n.d.). Math Curriculum Resource Project. Retrieved March 1, 2024, from https://ontariomath.support/?pg=results&type=subject&lang=EN&subject=Coding

Grade 3 – Coding Lesson – Dance patterns [Sample lesson plans]. (n.d.). Math Curriculum Resource Project. Retrieved March 1, 2024, from https://ontariomath.support/?pg=results&type=subject&lang=EN&subject=Coding

Grade 3 – Coding Lesson – Coding a table of values [Sample lesson plans]. (n.d.). Math Curriculum Resource Project. Retrieved March 1, 2024, from https://ontariomath.support/?pg=results&type=subject&lang=EN&subject=Coding

Grade 4 – Coding Lesson – Coding the way [Sample lesson plans]. (n.d.). Math Curriculum Resource Project. Retrieved March 1, 2024, from https://ontariomath.support/?pg=results&type=subject&lang=EN&subject=Coding

Grade 4 – Coding Lesson – Dance our way to coding using patterning [Sample lesson plans]. (n.d.). Math Curriculum Resource Project. Retrieved March 1, 2024, from https://ontariomath.support/?pg=results&type=subject&lang=EN&subject=Coding

Grade 4 – Coding Lesson – Coding on a coordinate grid [Sample lesson plans]. (n.d.). Math Curriculum Resource Project. Retrieved March 1, 2024, from https://ontariomath.support/?pg=results&type=subject&lang=EN&subject=Coding

Grade 4 – Coding Lesson – Let's paint our classroom [Sample lesson plans]. (n.d.). Math Curriculum Resource Project. Retrieved March 1, 2024, from https://ontariomath.support/?pg=results&type=subject&lang=EN&subject=Coding

Grade 4 – Coding Lesson – Scratch your probabilities! [Sample lesson plans]. (n.d.). Math Curriculum Resource Project. Retrieved March 1, 2024, from https://ontariomath.support/?pg=results&type=subject&lang=EN&subject=Coding

Grade 5 – Coding Lesson – Budgets and spreadsheets: Making plans [Sample lesson plans]. (n.d.). Math Curriculum Resource Project. Retrieved March 1, 2024, from https://ontariomath.support/?pg=results&type=subject&lang=EN&subject=Coding

Grade 5 – Coding Lesson – Navigational coding-unplugged [Sample lesson plans]. (n.d.). Math Curriculum Resource Project. Retrieved March 1, 2024, from https://ontariomath.support/?pg=results&type=subject&lang=EN&subject=Coding

Grade 5 – Coding Lesson – Recalling math facts through coding [Sample lesson plans]. (n.d.). Math Curriculum Resource Project. Retrieved March 1, 2024, from https://ontariomath.support/?pg=results&type=subject&lang=EN&subject=Coding

Grade 5 – Coding Lesson – Exploring patterns with fractions using Scratch [Sample lesson plans]. (n.d.). Math Curriculum Resource Project. Retrieved March 1, 2024, from https://ontariomath.support/?pg=results&type=subject&lang=EN&subject=Coding

Grade 6 – Coding Lesson – Part1: Getting ready to code transformations-unplugged [Sample lesson plans]. (n.d.). Math Curriculum Resource Project. Retrieved March 1, 2024, from https://ontariomath.support/?pg=results&type=subject&lang=EN&subject=Coding

Grade 6 – Coding Lesson – Part2: Coding transformations-unplugged [Sample lesson plans]. (n.d.). Math Curriculum Resource Project. Retrieved March 1, 2024, from https://ontariomath.support/?pg=results&type=subject&lang=EN&subject=Coding

Grade 6 – Coding Lesson – Part2: Mystery Integers! [Sample lesson plans]. (n.d.). Math Curriculum Resource Project. Retrieved March 1, 2024, from https://ontariomath.support/?pg=results&type=subject&lang=EN&subject=Coding

Grade 6 – Coding Lesson – Start small-making informed financial decisions [Sample lesson plans]. (n.d.). Math Curriculum Resource Project. Retrieved March 1, 2024, from https://ontariomath.support/?pg=results&type=subject&lang=EN&subject=Coding

Grade 6 – Coding Lesson – Coding and composite numbers [Sample lesson plans]. (n.d.). Math Curriculum Resource Project. Retrieved March 1, 2024, from https://ontariomath.support/?pg=results&type=subject&lang=EN&subject=Coding

Grade 7 – Coding Lesson – Get the wolf to her pups [Sample lesson plans]. (n.d.). Math Curriculum Resource Project. Retrieved March 1, 2024, from https://ontariomath.support/?pg=results&type=subject&lang=EN&subject=Coding

Grade 7 – Coding Lesson – Restaurant order app [Sample lesson plans]. (n.d.). Math Curriculum Resource Project. Retrieved March 1, 2024, from https://ontariomath.support/?pg=results&type=subject&lang=EN&subject=Coding

Grade 7 – Coding Lesson – $20 or 20% [Sample lesson plans]. (n.d.). Math Curriculum Resource Project. Retrieved March 1, 2024, from https://ontariomath.support/?pg=results&type=subject&lang=EN&subject=Coding

Grade 7 – Coding Lesson – Draw a maze [Sample lesson plans]. (n.d.). Math Curriculum Resource Project. Retrieved March 1, 2024, from https://ontariomath.support/?pg=results&type=subject&lang=EN&subject=Coding

Grade 7 – Coding Lesson – Transformations with coding [Sample lesson plans]. (n.d.). Math Curriculum Resource Project. Retrieved March 1, 2024, from https://ontariomath.support/?pg=results&type=subject&lang=EN&subject=Coding

Grade 7 – Coding Lesson – Coding and geometry [Sample lesson plans]. (n.d.). Math Curriculum Resource Project. Retrieved March 1, 2024, from https://ontariomath.support/?pg=results&type=subject&lang=EN&subject=Coding

Grade 8 – Coding Lesson – Coding with the Pythagorean theorem [Sample lesson plans]. (n.d.). Math Curriculum Resource Project. Retrieved March 1, 2024, from https://ontariomath.support/?pg=results&type=subject&lang=EN&subject=Coding

Grade 8 – Coding Lesson – Coding YouTube [Sample lesson plans]. (n.d.). Math Curriculum Resource Project. Retrieved March 1, 2024, from https://ontariomath.support/?pg=results&type=subject&lang=EN&subject=Coding

Gretter, S., & Yadav, A. (2016). Computational thinking and media & information literacy: An integrated approach to teaching twenty-first century skills. *TechTrends*, *60*, 510-516. https://doi.org/10.1007/s11528-016-0098-4

Grover, S. (2011, April). *Robotics and engineering for middle and high school students to develop computational thinking* [Conference presentation]. AERA 2011 Conference, New Orleans, LA, United States.

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 in school* (pp.19-38). Bloomsbury Academic. http://dx.doi.org/10.5040/9781350057142.ch-003

Hanna, J. L. (2008). A nonverbal language for imagining and learning: Dance education in K–12 curriculum. *Educational Researcher*, *37*(8), 491-506. https://doi.org/10.3102/0013189X083260

Hartley, J. (2004). Case study research. In C. Cassell & G. Symon (Eds.)., *Essential guide to qualitative methods in organizational research* (pp. 323-333). Sage.

Heinzman, E. (2022). "I LOVE MATH ONLY IF IT'S CODING": A case study of student experiences in an introduction to data science course. *Statistics Education Research Journal, 21*(2), 5:1-15. https://doi.org/10.52041/serj.v21i2.43

Herro, D., Quigley, C., Plank, H., & Abimbade. O. (2021) Understanding students' social interactions during making activities designed to promote computational thinking. *The Journal of Educational Research, 114*(2), 183-195. https://doi.org/10.1080/00220671.2021.1884824

Hoffmann, B. (2010). “I think I can, but I’m afraid to try”: The role of self-efficacy beliefs and mathematics anxiety in mathematics problem solving efficiency. *Learning and Individual Differences, 20*, 276–283. https://doi.org/10.1016/j.lindif.2010.02.001

Hwang, W. Y., Wang, C. Y., Hwang, G. J., Huang, Y. M., & Huang, S. (2008). A web-based programming learning environment to support cognitive development. *Interacting with Computers*, *20*(6), 524-534. https://doi.org/10.1016/j.intcom.2008.07.002

Iskrenovic-Momcilovic, O. (2020). Improving geometry teaching with scratch. *International Electronic Journal of Mathematics Education*, *15*(2), Article em0582. https://doi.org/10.29333/iejme/7807

Jiang, B., & Li, Z. (2021). Effect of Scratch on computational thinking skills of Chinese primary school students. *Journal of Computers in Education, 8*(4), 505-525. https://doi.org/10.1007/s40692-021-00190-z

Jones, S. P., Mitchell, B., & Humphreys, S. (2013). Computing at school in the UK. *CACM Report*. https://www.microsoft.com/en-us/research/wp content/uploads/2016/07/ComputingAtSchoolCACM.pdf

Julie, A. (2017, August 24). *Teaching coding in Canadian schools: How do the provinces measure up?* Global NEWS. https://globalnews.ca/news/3693932/teaching-coding-in-canadian-schools-how-do-the-provinces-measure-up/

Kalelioğlu, F. (2015). A new way of teaching programming skills to K-12 students: Code.org. *Computers in Human Behavior*, *52*, 200-210. https://doi.org/10.1016/j.chb.2015.05.047

Kalyenci, D., Metin, Ş., & Başaran, M. (2022). Test for assessing coding skills in early childhood. *Education and Information Technologies*, *27*, 4685-4708. https://doi.org/10.1007/s10639-021-10803-w

Kanbul, S., & Uzunboylu, H. (2017). Importance of Coding Education and Robotic Applications for Achieving 21st-Century Skills in North Cyprus. *International Journal of Emerging Technologies in Learning*, *12*(1), 130-140. https://doi.org/10.3991/ijet.v12i01.6097

Karagiannakis, G., Baccaglini-Frank, A., & Papadatos, Y. (2014). Mathematical learning difficulties subtypes classification. *Frontiers in Human Neuroscience*, *8*, Article 57. https://doi.org/10.3389/fnhum.2014.00057

Ke, F. (2014). An implementation of design-based learning through creating educational computer games: A case study on mathematics learning during design and computing. *Computers & Education*, *73*, 26-39.https://doi.org/10.1016/j.compedu.2013.12.010

Khuraisah, M. N., Khalid, F., & Husnin, H. (2020). Preparing graduates with digital literacy skills toward fulfilling employability need in 4IR Era: A review. *International Journal of Advanced Computer Science and Applications*, *11*(6). 307-316. https//doi.org/10.14569/IJACSA.2020.0110641

Kotsopoulos, D., Floyd, L., Khan, S., Namukasa, I. K., Somanath, S., Weber, J., & Yiu, C. (2017). A pedagogical framework for computational thinking. *Digital Experiences in Mathematics Education*, *3*(2), 154-171. https://doi.org/10.1007/s40751-017-0031-2

Kong, R., & Wong, G. K. (2017, December). Teachers' perception of professional development in coding education. *Proceedings of 2017 IEEE 6th International Conference on*

Teaching, Assessment, and Learning for Engineering (TALE), HK, 377-380. https:// doi: 10.1109/TALE.2017.8252365.

Lafee, S. (2017). Coding: The new 21st-century literacy?. *The Education Digest*, *83*(2), 25-30.

Lee, J., & Junoh, J. (2019). Implementing unplugged coding activities in early childhood classrooms. *Early Childhood Education Journal*, *47*(6), 709-716. https://doi.org/10.1007/s10643-019-00967-z

Lengel, T., & Kuczala, M. (Eds.). (2010). *The kinesthetic classroom: Teaching and learning through movement*. Corwin Press.

Leonard, J., Buss, A., Gamboa, R., Mitchell, M., Fashola, O. S., Hubert, T., & Almughyirah, S. (2016). Using robotics and game design to enhance children's self-efficacy, STEM attitudes, and computational thinking skills. *Journal of Science Education and Technology*, 25, 860-876. https://doi.org/10.1007/s10956-016-9628-2

Levin, B. (2008, May 15-18). *Thinking about knowledge mobilization* [Symposium technical report]. An invitational symposium sponsored by the Canadian Council on Learning and the Social Sciences and Humanities Research Council of Canada, Harrison Hot Springs, BC, Canada, 1-14. http://en.copian.ca/library/research/ccl/levin_report_en/levin_report_en.pdf

Mäntylä, M. V., Graziotin, D., & Kuutila, M. (2018). The evolution of sentiment analysis—A review of research topics, venues, and top cited papers. *Computer Science Review*, *27*, 16-32. https://doi.org/10.1016/j.cosrev.2017.10.002

Marinus, E., Powell, Z., Thornton, R., McArthur, G., & Crain, S. (2018). Unravelling the cognition of coding in 3-to-6-year-olds: The development of an assessment tool and the relation between coding ability and cognitive compiling of syntax in natural language. *Proceedings of the 2018 ACM Conference on International Computing Education Research.* 133-141. https://doi.org/10.1145/3230977.3230984

Mason, S. L., & Rich, P. J. (2019). Preparing elementary school teachers to teach computing, coding, and computational thinking. *Contemporary Issues in Technology and Teacher Education*, *19*(4), 790-824.

Mason, S. L., & Rich, P. J. (2020). Development and analysis of the Elementary Student Coding Attitudes Survey. *Computers & Education, 153,* Article 103898. https://doi.org/10.1016/j.compedu.2020.103898

Merriam, S. B. (2002). Introduction to qualitative research. *Qualitative research in practice: Examples for discussion and analysis*, *1*(1), 1-17.

Miller, J. (2019). STEM education in the primary years to support mathematical thinking: Using coding to identify mathematical structures and patterns. *ZDM*, *51*(6), 915-927. https://doi.org/10.1007/s11858-019-01096-y

Ministry of Education of Ontario. (2020). *Curriculum and Resources: Mathematics (2020 Edition)*. https://www.dcp.edu.gov.on.ca/en/curriculum/elementary-mathematics

Ministry of Education of the People's Public of China. (2022). *The Ministry of Education issued the compulsory education curriculum plan and Notice of Curriculum Standards (2022 Edition)*.

Mishler, E. (1979). Meaning in context: Is there any other kind?. *Harvard Educational Review*, *49*(1), 1-19. https://doi.org/10.17763/haer.49.1.b748n4133677245p

Noone, M., & Mooney, A. (2018). Visual and textual programming languages: a systematic review of the literature. *Journal of Computers in Education*, *5*(2), 149-174. https://doi.org/10.1007/s40692-018-0101-5

Ontario. (2021, June). *Support your child's math learning*. CA. Ontario. Ministry of Education. https://www.ontario.ca/page/support-your-childs-math-learning

Ontario. (2022). *Mathematics.* CA. Ontario. Ministry of Education. https://www.dcp.edu.gov.on.ca/en/curriculum/elementary-mathematics

Ontario. (2022). *Mathematics.* Curriculum and Resources, C3. Coding (Grade 1) [Program of studies]. https://www.dcp.edu.gov.on.ca/en/curriculum/elementary-mathematics/grades/g1-math/strand-c/c3

Ontario. (2022). *Mathematics.* Curriculum and Resources, C3. Coding (Grade 2) [Program of studies]. https://www.dcp.edu.gov.on.ca/en/curriculum/elementary-mathematics/grades/g2-math/strand-c/c3

Ontario. (2022). *Mathematics.* Curriculum and Resources, C3. Coding (Grade 3) [Program of studies]. https://www.dcp.edu.gov.on.ca/en/curriculum/elementary-mathematics/grades/g3-math/strand-c/c3

Ontario. (2022). *Mathematics.* Curriculum and Resources, C3. Coding (Grade 4) [Program of studies]. https://www.dcp.edu.gov.on.ca/en/curriculum/elementary-mathematics/grades/g4-math/strand-c/c3

Ontario. (2022). *Mathematics.* Curriculum and Resources, C3. Coding (Grade 5) [Program of studies]. https://www.dcp.edu.gov.on.ca/en/curriculum/elementary-mathematics/grades/g5-math/strand-c/c3

Ontario. (2022). *Mathematics.* Curriculum and Resources, C3. Coding (Grade 6) [Program of studies]. https://www.dcp.edu.gov.on.ca/en/curriculum/elementary-mathematics/grades/g6-math/strand-c/c3

Ontario. (2022). *Mathematics.* Curriculum and Resources, C3. Coding (Grade 7) [Program of studies]. https://www.dcp.edu.gov.on.ca/en/curriculum/elementary-mathematics/grades/g7-math/strand-c/c3

Ontario. (2022). *Mathematics.* Curriculum and Resources, C3. Coding (Grade 8) [Program of studies]. https://www.dcp.edu.gov.on.ca/en/curriculum/elementary-mathematics/grades/g8-math/strand-c/c3

Ontario. (2023, March 1st). In *Wikipedia*. https://en.wikipedia.org/wiki/Ontario

Ontario. (2024, February). *What we do*. CA. Ontario. Ministry of Education. https://www.ontario.ca/page/ministry-education

Ontario Math Support. (2024 March). *Math Curriculum Resource Project*. Ontario Math Support. https://ontariomath.support/

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

Piaget, J. (1953). *The origins of intelligence in the child*. Routledge & Kegan Paul Ltd.

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

Posner, G. (1992/2004). *Analyzing the curriculum*. McGraw Hill.

Prentice, M. G. (1971). *Theory-based paradigms for the generation of curricular designs*. The Ohio State University.

Resnick, M., & Rusk, N. (2020). Coding at a crossroads. *Communications of the ACM*, *63*(11), 120-127. https://doi.org/10.1145/3375546

Rich, K. M., Yadav, A., & Schwarz, C. V. (2019). Computational thinking, mathematics, and science: Elementary teachers' perspectives on integration. *Journal of Technology and Teacher Education*, *27*(2), 165-205.

Saxena, A., Lo, C. K., Hew, K. F., & Wong, G. K. W. (2020). Designing unplugged and plugged activities to cultivate computational thinking: An exploratory study in early childhood education. *The Asia-Pacific Education Researcher, 29*(1), 55–66. https://doi.org/10.1007/s40299-019-00478-w

Seow, P., Looi, CK., How, ML., Wadhwa, B., & Wu, LK. (2019). Educational Policy and Implementation of Computational Thinking and Programming: Case Study of Singapore. In SC. Kong & H. Abelson (Eds.), *Computational Thinking Education.* Springer. https://doi.org/10.1007/978-981-13-6528-7_19

Sheridan, I., Goggin, D., & O'Sullivan, L. (2016). Exploration of learning gained through coderdojo coding activities. *Proceedings of INTED2016, Spain,* 6541-6548.

Shiel, G., Cregan, Á., McGough, A., & Archer, P. (2012). O*ral language in early childhood and primary education (3-8 years).* Dublin: National Council for Curriculum and Assessment.

https://www.erc.ie/documents/oral_language_in_early_childhood_and_primary_education_3-8_years_.pdf

Shkedi, A. (2009). From curriculum guide to classroom practice: Teachers' narratives of curriculum application. *Journal of Curriculum Studies*, *41*(6), 833-854. https://doi.org/10.1080/00220270902927030

Shulman, L. S. (1986). Those who understand: Knowledge growth in teaching. *Educational Researcher, 15*(2), 4-14. https://doi.org/10.2307/1175860

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

Sneck, S., Viholainen, H., Syväoja, H., Kankaapää, A, Hokonen, H., Poikkeus, AM & Tammelin, T. Effects of school-based physical activity on mathematics performance in children: a systematic review. *International Journal of Behavioral Nutrition and Physical Activity. 16*, Article 109. https://doi.org/10.1186/s12966-019-0866-6

Stigberg, H., & Stigberg, S. (2020). Teaching programming and mathematics in practice: A case study from a Swedish primary school. *Policy Futures in Education*, *18*(4), 483-496. https://doi.org/10.1177/1478210319894785

Strawhacker, A., & Bers, M. U. (2018). Promoting positive technological development in a Kindergarten makerspace: A qualitative case study. *European Journal of STEM Education*, *3*(3), Article 9. https://doi.org/10.20897/ejsteme/3869

Strawhacker, A., & Bers, M. U. (2019). What they learn when they learn coding: Investigating cognitive domains and computer programming knowledge in young children. *Educational Technology Research and Development*, *67*, 541-575. https://doi.org/10.1007/s11423-018-9622-x

Thornton Moore, D. (2004). Curriculum at work: An educational perspective on the workplace as a learning environment. *Journal of Workplace Learning*, *16*(6), 325-340. https://doi.org/10.1108/13665620410550303

Tugun, V., Uzunboylu, H., & Ozdamli, F. (2017). Coding education in a flipped classroom. *TEM Journal*, *6*(3). 599-606. https//doi.org/10.18421/TEM63-23

Tuomi, P., Multisilta, J., Saarikoski, P., & Suominen, J. (2018). Coding skills as a success factor for a society. *Education and Information Technologies*, *23*, 419-434. https://doi.org/10.1007/s10639-017-9611-4

User experience design. (2024, March 10). In *Wikipedia*. https://en.wikipedia.org/wiki/User_experience_design

Van Roy, P., & Haridi, S. (2004). *Concepts, techniques, and models of computer programming*. MIT Press.

Videnovik, M., Vlahu-Gjorgievska, E., & Trajkovik, V. (2021). To code or not to code: Introducing coding in primary schools. *Computer Applications in Engineering Education, 29*(5), 1132-1145. https://doi.org/10.1002/cae.22369

Vogler, J. S., Thompson, P., Davis, D. W., Mayfield, B. E., Finley, P. M., & Yasseri, D. (2018). The hard work of soft skills: augmenting the project-based learning experience with

interdisciplinary teamwork. *Instructional Science*, *46*, 457-488. https://doi.org/10.1007/s11251-017-9438-9

Vukovic, R. K., Kieffer, M. J., Bailey, S. P., & Harari, R. R. (2013). Mathematics anxiety in young children: Concurrent and longitudinal associations with mathematical performance. *Contemporary Educational Psychology, 38*, 1–10. https://doi.org/10.1007/BF03174765

Wang, M. C., Haertel, G. D., & Walberg, H. J. (1997). Fostering educational resilience in inner-city schools. In H. J. Walberg, O. Reyes, & R. P. Weissberg (Eds.), *Children and youth: Interdisciplinary perspectives* (pp. 119–140). Sage Publications, Inc.

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

Welch, L., Kozlowski, J., & Evans, H. (2019, April 10). *Coding to develop early mathematical and computational thinking in kindergarten: A case study* [Oral Presentation]. SRS 2019 Student Research Symposium, Logan, UT, United States.

Welch, L. E., Shumway, J. F., Clarke-Midura, J., & Lee, V. R. (2022). Exploring measurement through coding: Children's conceptions of a dynamic linear unit with robot coding toys. *Education Sciences*, *12*(2), 143. https://doi.org/10.3390/educsci12020143

Wing, J. M. (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. https://people.cs.vt.edu/~kafura/CS6604/Papers/CT-What-And-Why.pdf

Yadav, A., Stephenson, C., & Hong, H. (2017). Computational thinking for teacher education. *Communications of the ACM*, *60*(4), 55-62. https://doi.org/10.1145/2994591

Yin, R. K. (2003). *Case study research: Design and methods* (3rd ed.). Sage Publications.

Young, M. (2014). What is a curriculum and what can it do?. *Curriculum Journal*, *25*(1), 7-13. https://doi.org/10.1080/09585176.2014.902526