A Thematic Analysis Exploring Flexibility in Programming-based Mathematical Problem Solving

Huiyan YE[a], Oi-Lam NG[a*] & Zhihao CUI[a]
[a]*Faculty of Education, The Chinese University of Hong Kong, China*
* oilamn@cuhk.edu.hk

Abstract: Research on computational thinking (CT) have received significant attention in recent years, but how CT and mathematical thinking (MT) work together to influence problem solver's computational strategies and competency has not been illuminated. Taking the method of thematic analysis combined with a conceptual framework of flexibility in programming-based mathematical problem solving, this study aims to shed light on the interplay between CT and MT. We identify two kinds of flexibility when a problem solver engages in programming-based mathematical problem-solving: translation flexibility and integrated flexibility. We hope that this research will contribute to a deeper understanding of the interplay of CT and MT as a basis for conceptualizing an interdisciplinary form of STEM education and thinking on task design, pedagogy, and assessment approaches of CT-based mathematics instruction.

Keywords: flexibility, computational thinking, mathematical thinking, interplay

1. Introduction

Computational thinking (CT) is regarded as a fundamental skill required in daily life, and a kind of analytical thinking that shares common characteristics with mathematical thinking (MT) in many ways (Wing, 2011). Thus, researchers have investigated ways to integrate CT into mathematics teaching and learning (e.g., Ng & Cui, 2021). However, one of the key challenges is to understand how CT and MT work together to influence students' strategies when engaging in mathematical problem-solving in computational contexts. Previous studies have identified discrepancies between CT and MT, which make mathematical problem-solving in computational contexts more challenging for students (Cui & Ng, 2021). Therefore, understanding the complex relationship of CT and MT is vitally important if CT is to be meaningfully integrated into mathematics education.

While the breadth of research has advanced the empirical basis of understanding what CT-based mathematics instruction entails, very little is currently known about the processes of how CT and MT specifically work together in this context. Meanwhile, noting the diversity of solution strategies in the CT-based mathematical problem-solving, research on flexibility in mathematical problem-solving may provide possible directions for our attempts to explore the interplay between CT and MT. Flexibility refers to "the ability to change according to particular circumstances" (Star & Seifert, 2006, p. 281), which relies on the problem solver's understanding of the knowledge relevant to the problem situation. If the problem context is extended to a CT-based mathematical problem, not only would multiple programming solution strategies be possible, but the problem solvers would also require knowledge of programming and mathematics, as well as to adapt this knowledge according to the programming environment used and mathematical ideas involved respectively within the problem context. In this study, we are interested in delving into this very interplay between CT and MT from perspective of flexibility. Therefore, the study is guided by the research question: What types of flexibility are exhibited during CT-based mathematical problem-solving?

2. Theoretical Background

2.1 Computational Thinking and Mathematical Thinking

In this study, we define CT as "the thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent" (Wing, 2011, p. 20), and MT not as "thinking about the subject matter of mathematics but a style of thinking that is a function of particular operations, processes, and dynamics recognizably mathematical" (Burton, 1984, p. 35). In other words, both CT and MT refer to the cognitive structures that govern an individual's programming and mathematical activities respectively, as well as the anticipation of the results of those activities. CT is thought to share a similar approach to MT in problem solving (Wing, 2011). Researchers also attempted to find out some of the empirical connections between CT and MT, with some reckoning that CT and MT are mutually supportive or have overlapping features. For example, Ng and Cui (2021) argued that mathematical exploration can serve as a site that allows students to engage in CT in meaningful ways. On the other hand, the presence of CT and MT may occur in mismatching ways. As Rich and colleagues (2018) point out, mathematics focuses on students' flexible arithmetic operations and arithmetic strategies, while CT focuses on sequence so that the computer can recognize and execute. While these studies point to an inter-relationship between CT and MT (e.g., CT support MT, or vice versa, or the two are constraining), the conclusions are relatively broad and lack a detailed perspective on how such facilitative or hindering relationships of CT and MT arise and develop. That is, there requires more evidence to capture the presence or expression of CT and MT in a clear way and how they affect students' thinking processes when they engage in CT-based mathematical problem solving.

2.2 Flexibility in Problem Solving

Flexibility is considered to be a key competency in problem solving. Even if research focusing on flexibility has been studied extensively in different domains of mathematics, there has been little research on interdisciplinary flexibility, that is, how flexibility manifest across different subject domains. In particular, the flexibility of working across CT and MT in the context of CT-based mathematical problem-solving not yet been empirically studied and analysed. In this study, we adopt flexibility as "knowledge of multiple strategies and the ability to select the most appropriate strategy for a given problem and problem-solving circumstances" (Star et al., 2022, p. 2) so that we can observe how flexibility are demonstrated through various tasks in our designed CT-based mathematical problems. This investigation can potentially refine our understanding of how problem solvers make connections between computational and mathematical concepts. In particular, we pay attention to the tasks in which problem solvers are situated to translate between mathematical and programming language or integrate mathematics concepts and computational concepts while making meaningful connections of CT and MT. When analysing these tasks, we recognise that programming itself is a tool-mediated activity; therefore, the instrumentation of different programming tools would necessitate different kinds of CT-MT translations and connections to occur. For this reason, we choose to address one specific programming language, Scratch.

3. Methodology

3.1 Context and Participant

This study is part of a larger design-based research addressing the research objectives of designing a series of tasks for supporting computationally enhanced mathematical problem-solving in school mathematics classrooms, and of studying student learning trajectories in terms of developing their CT, MT, and problem-solving competence in the designed tasks. The participants in this study were from two programming-based mathematical problem-solving programs. One was a student program, which was design-based research (DBR) undertaken with three iterative cycles of implementations with seventh to ninth grade (age 12 to 15) students (n = 95) from three secondary schools in Hong Kong. Another one was a teacher program, which adopted task-based semi-structured interview to collect data. Eight in-service mathematics teachers from different schools in Mainland China, participated in a five-week online professional development workshop in Scratch programming. The two programs were conducted independently upon obtaining participants consent, with the student program being undertaken in 2021 and the teacher program being conducted in 2022 via Zoom.

In this study, we adopt the method of thematic analysis to further interpret the translation/connection involved between CT and MT in the programming solutions to the chosen tasks through the lens of flexibility. As the purpose of this study was to explore the flexibility of learners' strategies in solving programming-based mathematics problems, although participants included two different groups (i.e., students and in-service mathematics teachers), we believe our data still appropriate to answer the RQs, since the programming solutions, as opposed to the participants (or problem solvers) were the focus of this study.

3.2 Data Collection and Analysis

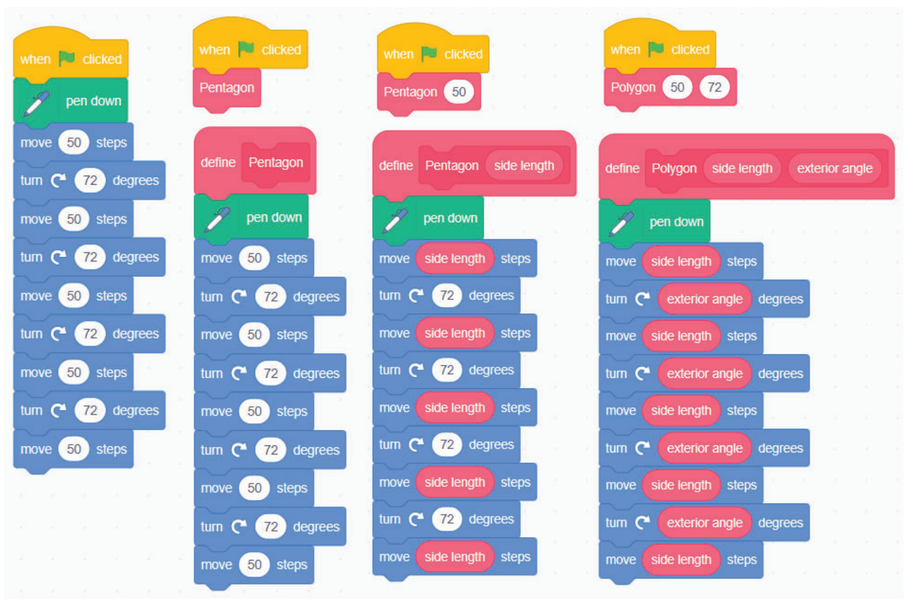
Each program contained an introduction game and several tasks related to four domains: number, algebra, geometry, and data handling. A total of 18 tasks were studied in response to the research question posed, in which students program contained 14 tasks (1 introduction game, 5 in number and algebra, 4 in geometry, 4 in data handling) and teachers program contained 4 tasks with one task in each topic. All tasks are programming-based mathematical problems and are all open-ended problems with no fixed coding strategy. For example, "Please write a code project that can be used to verify whether an entered number is prime or not." Coding-related solutions are all thought out and collaborated by participants (students or teachers). The researchers will observe participants' problem-solving process and guide the participants to reflect through task-based interviews at the appropriate time, which aims to understand what they think and to help them solve the difficulties they encounter during the programming process. Therefore, our data include (a) video and voice recording of interactions and interview between researchers and participants; (b) collection of field notes, screen capturing, developed artefacts and documents in both programs, which help us to gain a deep understanding of the research questions.

As a qualitative analysis method to identify, analyze and report patterns of data (Braun & Clarke, 2006), thematic analysis is regarded as an effective approach for identifying similarities and differences, yielding unforeseen insights and summarizing key features of a large data set (Nowell et al., 2017). Therefore, we adopted thematic analysis with the aim to identify how participants exhibited flexibility in their programming solutions for mathematical problems. Taking our designed tasks and participants programming solutions as the unit of analysis, we conducted latent thematic analysis, which developing the themes necessitates interpretive effort and the resulting analysis is not just descriptive, but is already theorized (Braun & Clarke, 2006). Our analysis was implemented following the six steps of doing thematic analysis outlined by Braun and Clarke (2006).

4. Results

By analyzing and generalizing participants' problem-solving strategies in solving CT-based mathematical problems, the results of the thematic analysis yielded two main kinds of flexibility (i.e., translation flexibility and integrated flexibility) and six sub-categories. Translational flexibility refers to a cross-disciplinary, cross-problem context capability that involves translation between mathematical and programming language, with the two processes being independent. In contrast, integrated flexibility is an overall capability similar to flexibility in a mathematical problem context (e.g., to solve an algebra equation; Star & Seifert, 2006). It is the flexibility to solve problems in a programming environment that involves the use of knowledge from both disciplines, with the MT and CT realizations being inseparable as a whole. Due to the length of the article, we will only cover integrated flexibility in detail with a geometry example.

One of the most evident examples of integrated flexibility is the setup of subroutines, often known as creating *My Blocks* in Scratch. As an important CT concept, subroutines can be divided into two types. One is the parameterless subroutine, which calls an entire piece of code directly and repeatedly as a whole, without altering it before or after the subroutine function is invoked, but rather given it a separate name for wrapping (Figure 1a-b). This type of subroutine does not require programmer to make any adjustments to the codes. Instead of executing the original code directly, another kind of subroutine with parameters requires programmers to comprehend the section of codes in its entirety, identify the key attribute and numerical relationships between them, and then modify the codes by abstracting parameters to be a subroutine (Figure 1c). In this section, we will focus on subroutines with parameters and the related integrated flexibility will be examined through a geometry drawing task.



(a)
```
when 🏁 clicked
pen down
move 50 steps
turn ↻ 72 degrees
move 50 steps
turn ↻ 72 degrees
move 50 steps
turn ↻ 72 degrees
move 50 steps
turn ↻ 72 degrees
move 50 steps
```

(b)
```
when 🏁 clicked
Pentagon

define Pentagon
pen down
move 50 steps
turn ↻ 72 degrees
move 50 steps
turn ↻ 72 degrees
move 50 steps
turn ↻ 72 degrees
move 50 steps
turn ↻ 72 degrees
move 50 steps
```

(c)
```
when 🏁 clicked
Pentagon 50

define Pentagon side length
pen down
move side length steps
turn ↻ 72 degrees
move side length steps
turn ↻ 72 degrees
move side length steps
turn ↻ 72 degrees
move side length steps
turn ↻ 72 degrees
move side length steps
```

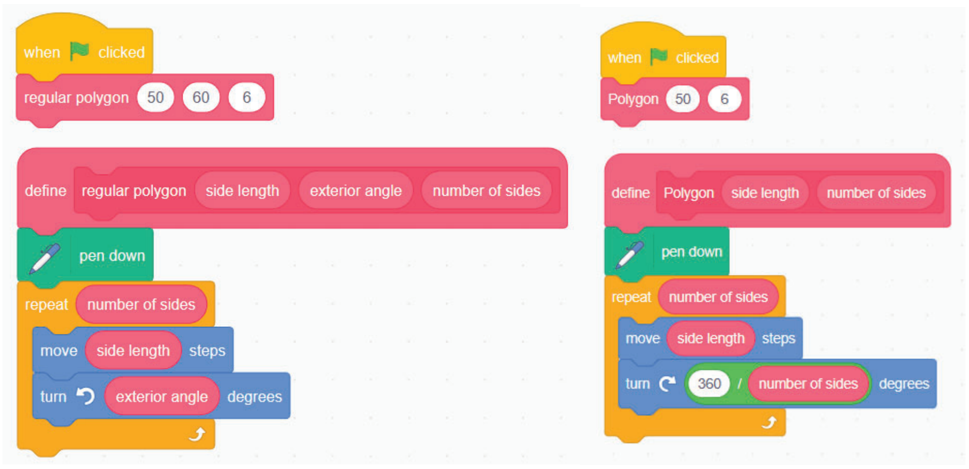
(d)
```
when 🏁 clicked
Polygon 50 72

define Polygon side length exterior angle
pen down
move side length steps
turn ↻ exterior angle degrees
move side length steps
turn ↻ exterior angle degrees
move side length steps
turn ↻ exterior angle degrees
move side length steps
turn ↻ exterior angle degrees
move side length steps
```

(a) (b) (c) (d)

Figure 1. Setting of My Blocks as Subroutine in Scratch

As we know, to draw a regular polygon in paper-and-pencil context, we first need to know the polygon's side length, the degree of interior angles, then to draw it with a ruler and a protractor. While to achieve this mathematical process in Scratch, the Pen function is used to record the trajectory of a Sprite. For example, to draw a regular pentagon with side length of 50, the pen needs to [move 50 steps, turn clockwise 72 degrees] then repeat this process five times (Figure 1a). The code at this point does not yet make use of subroutine, but if programmers try to use *My Block* to simplify the programming process or design a program that can change the size and shape of the polygon at any time, mathematical consideration is required for setting parameters in the subroutine (Figure 1c-d).

It is worth noting that the process of changing the original codes (Figure 1a) to a piece of code with subroutine (Figure 1c-d; Figure 2a-b) requires the involvement of both MT and CT. The interplay between CT and MT is evident in this integration, particularly in the different strategies from Figure 1c-d to Figure 2a-b. For example, programmers can abstract a parameter "side length" in the *My Block* and replace all the numbers or variables in the source code that represent the length of the sides to create a subroutine served to control the size of a pentagon (Figure 1c). Here, we can see that such a change is mediated by both CT and MT because one may have to figure out which variable represent the side length that can be a parameter in the subroutine. CT support programmers to create the subroutine, while MT helps determine the first parameter in the subroutine, i.e., the size of the polygon is determined by the length of the sides. However, when the programmer went on to optimize the subroutine so that it can also change the shape of the polygon, it does not mean that mathematical property “different polygons have different interior angles” makes it possible to add a second parameter “exterior angle” directly to the *My Block* in Figure 1c. Otherwise, even if the second parameter “exterior angle” is added to the *My Block* (Figure 1d), it will not be possible to draw any other regular polygon than a regular pentagon, because only with the parameter “polygon 50 72” will the matching angle and number of sides can form an enclosed polygon. This is because the number of sides and the degree of interior angles determine the shape of the polygon, while the subroutine in Figure 1c only contains codes of [move 50 steps, turn clockwise 72 degrees] five times which does not have the function to change number of sides that follow the change of degree of exterior angle. The key to solve such problems is the effective coordination of CT and MT.



```
when 🏁 clicked
regular polygon (50) (60) (6)

define regular polygon (side length) (exterior angle) (number of sides)
pen down
repeat (number of sides)
    move (side length) steps
    turn ↺ (exterior angle) degrees
```

```
when 🏁 clicked
Polygon (50) (6)

define Polygon (side length) (number of sides)
pen down
repeat (number of sides)
    move (side length) steps
    turn ↻ ((360) / (number of sides)) degrees
```

(a) (b)

Figure 2. Setting of My Blocks as Subroutine

Furthermore, although the subroutine in Figure 2a with three parameters (i.e., “side_length”, “exterior_angle”, “number_of_sides”) can change the size and shape of the regular polygon, there actually exists a specific quantitative relationship between the degree of exterior angle and the number of sides, i.e., degree of exterior angle = 360/number of sides, so the *My Block* in Figure 2a can be optimised to a subroutine with only two parameters affecting the shape (number_of_sides) and size (side_length) of the polygon in Figure 2b. This reveals the synergy between CT and MT in the process of setting up subroutines with parameters. Thus, we can observe that during the process of setting subroutines, the key concepts of CT (subroutines, loops) and MT (mathematical properties) constrain and facilitate each other: When problem solvers want to set up flexible subroutines that can change the shape and size of regular polygons, CT concepts such as subroutines and loops play a key role in supporting the adaptation of the program, while quantitative relationships in mathematics involved in MT responsible for decision making on parameters setting in the subroutine, the choice of using which CT concepts and how they are used. That is, the number of sides determine the number of loops; the parameters of the subroutines are the attributes that control size and shape—the length and the number of sides, respectively. It is in the combined effect of CT and MT that provide opportunity for programmers to abstract the key parameter that determines the shape of a regular polygon. We have named such an integrative programming process as “integrated flexibility” in which CT and MT co-exist and support each other.

5. Discussion

Overall, translation flexibility is a cross-disciplinary, cross-problem context competence, while integrated flexibility is a holistic capability similar to flexibility in mathematical problem contexts (Star et al., 2022; Star & Seifert, 2006), that is, the general flexibility to solve problems in a programming environment that includes knowledge from both mathematics and programming disciplines. It involves a complex interaction between MT and CT which different from the one-way conversion of translation flexibility. In general, integrated flexibility is more often found in the process of selecting a CT technique when the problem solver has a clear mathematical solution. It is vital to take into account which mathematical approach and which CT technique are compatible. If the two do not match, adjustments must be made to the mutual circumstance in order to develop an effective programming solution.

We believe such a conceptual framework has potential for understanding the mutual cognitive development in CT and MT, including how mathematical concepts co-emerge with programming processes in a problem-solving context, and the role of CT in such processes. We understand that because programming is a tool-mediated activity, different programming tools will be instrumented differently, necessitating various CT-MT translations and interactions. It is hoped that this research will contribute to a deeper understanding of the interplay of CT and MT as a basis for conceptualizing an interdisciplinary form of STEM education and thinking on task design, pedagogy, and assessment approaches of CT-based mathematics instruction. Moreover, the findings will offer theoretical contributions that inform researchers and mathematics educators in terms of student learning characteristics during CT-based mathematics instruction. Finally, we have noticed that our data from two different participant groups exhibit some comparable features of flexibility, which may point to a certain universality of problem solvers' flexibility in programming-based mathematical problems. As a concluding remark of the paper, we pose several questions that arise from our exploration—Given the existence of multiple solutions, are the prevailing programming solutions simpler or more complex to solve the problem? Why are these approaches more common? What factors are relevant to them? How are the problem solvers' flexibility to use different types of translations affected by their CT and MT? We call for more research to consider the different kinds of flexibility demonstrated in CT-based mathematical problem solving to sketch the big picture of how CT and MT emerge in complementary or mismatching ways.

References

Braun, V., & Clarke, V. (2006). Using thematic analysis in psychology. *Qualitative Research in Psychology*, *3*(2), 77–101.

Burton, L. (1984). Mathematical thinking: The Struggle for meaning. *Journal for Research in Mathematics Education*, *15*(1), 35–49. https://doi.org/10.2307/748986

Cui, Z., & Ng, O. (2021). The interplay between mathematical and computational thinking in primary school students' mathematical problem-solving within a programming environment. *Journal of Educational Computing Research*, *59*(5), 988–1012.

Nowell, L. S., Norris, J. M., White, D. E., & Moules, N. J. (2017). Thematic analysis: Striving to meet the trustworthiness criteria. *International Journal of Qualitative Methods*, *16*(1), 1609406917733847.

Ng, O., & Cui, Z. (2021). Examining primary students' mathematical problem-solving in a programming context: Towards computationally enhanced mathematics education. *ZDM – Mathematics Education*, *53*(4), 847–860.

Rich, K. M., Strickland, C., Binkowski, T. A., Moran, C., & Franklin, D. (2018). K-8 learning trajectories derived from research literature: Sequence, repetition, conditionals. *ACM Inroads*, *9*(1), 46–55.

Star, J. R., & Seifert, C. (2006). The development of flexibility in equation solving. *Contemporary Educational Psychology*, *31*(3), 280–300.

Star, J. R., Tuomela, D., Joglar-Prieto, N., Hästö, P., Palkki, R., Abánades, M. Á., Pejlare, J., Jiang, R. H., Li, L., & Liu, R.-D. (2022). Exploring students' procedural flexibility in three countries. *International Journal of STEM Education*, 9(1), 4.

Wing, J. (2011). Research notebook: Computational thinking—What and why. *The Link Magazine*, *6*, 20–23.

Ye, H., Liang, B., Ng, O., & Chai, C. S. (2023). Integration of computational thinking in K-12 mathematics education: A systematic review on CT-based mathematics instruction and student learning. *International Journal of STEM Education*. Doi:10.1186/s40594-023-00396-w