How teachers practice Computational Thinking assessment in Mathematics classrooms

AINO UKKONEN, KATARINA PAJCHEL, CONSTANTINOS XENOFONTOS

Oslo Metropolitan University (OsloMet)
Norway
aino.ukkonen@oslomet.no
katarina.pajchel@oslomet.no

University of Agder
Norway
konstantinos.xenofontos@uia.no

Abstract

The integration of computational thinking (CT) into school mathematics presents both pedagogical opportunities and assessment challenges. This case study investigates how two Norwegian mathematics teachers, Stephen and Lena, practice CT assessment in primary and lower secondary classrooms. Drawing on classroom observations and interviews, the study explores how formative assessment through feedback supports student learning during CT-integrated mathematics lessons. The findings reveal that debugging, decomposition, abstraction, and algorithms are central to CT assessment but are interpreted and enacted differently by the two teachers. Stephen employed structured tools such as decomposition checklists and feedback on functions and variables to scaffold problem solving, while Lena used exploratory reasoning, productive failure, and geometric representations to elicit student thinking. Both used feedback to bridge CT constructs and mathematical understanding, and a significant finding in this study is that CT feedback and assessment in mathematics are deeply intertwined, reflecting the integration of CT and mathematics. The study highlights the importance of strategic feedback and the need for clearer frameworks connecting CT and mathematics in assessment.

Keywords

Computational Thinking, formative assessment, feedback, Mathematics teachers, assessment practices

Résumé

L'intégration de la pensée computationnelle (PC) dans l'enseignement des mathématiques à l'école offre à la fois des opportunités pédagogiques et pose des défis en matière d'évaluation. Cette étude de cas examine la manière dont deux professeurs de mathématiques norvégiens, Stephen et Lena, mettent en œuvre l'évaluation de PC dans leurs classes du primaire et du premier cycle du secondaire. S'appuyant sur des observations en classe et des entretiens, cette étude examine comment l'évaluation formative, par le biais du retour d'information, favorise l'apprentissage des élèves lors de cours de mathématiques intégrant la PC. Les résultats montrent que le débogage, la décomposition, l'abstraction et les algorithmes occupent une place centrale dans l'évaluation de PC, mais qu'ils sont interprétés et mis en pratique différemment par les deux enseignants. Stephen a utilisé des outils structurés, tels que des listes de contrôle pour la décomposition et des feedbacks sur les fonctions et les variables, afin de faciliter la résolution de problèmes, tandis que Lena a eu recours au raisonnement exploratoire, à l'échec constructif et aux représentations géométriques pour faire ressortir la réflexion des élèves. Les deux ont utilisé le feedback pour faire le lien entre les concepts de la PC et la compréhension des mathématiques ; l'une des conclusions majeures de cette étude est que le feedback et l'évaluation en matière de PC et de mathématiques sont étroitement liés, ce qui reflète l'intégration de la PC et des mathématiques. Cette étude souligne l'importance d'un feedback stratégique et la nécessité de disposer de cadres plus clairs établissant un lien entre la PC et les mathématiques dans l'évaluation.

Mots-clés

Pensée computationnelle, évaluation formative, feedback, professeurs de mathématiques, pratiques d'évaluation

Cite this article

Ukkonen, A., Pajchel, K., & Xenofontos, C. (2026). How teachers practice Computational Thinking assessment in Mathematics classrooms. *Review of Science, Mathematics and ICT Education*, *20*(1), 25-57. https://doi.org/10.26220/rev.5572

Introduction

The influential role of technology, programming, and problem solving in society is reflected in the global introduction of computational thinking (CT) into compulsory education (Bocconi et al., 2022; So et al., 2020; Yadav et al., 2016). Although CT can be developed across school subjects (Mueller et al., 2017), mathematics has become a primary arena for its integration in many European countries, alongside growing connections with science and the arts (Bocconi et al., 2022; Pajchel et al., 2024). Embedding CT in mathematics, however, presents both pedagogical and assessment challenges (Nordby et al.,

2022b; Ukkonen, Pajchel, et al., 2024; Tang et al., 2020). Teachers often find it difficult to link CT meaningfully to mathematical content and tend to view it as an external addition rather than as a natural extension of mathematics (Nordby et al., 2022b). They also express uncertainty about how CT should be assessed, partly because its relationship to programming and mathematics learning remains conceptually unclear (Ukkonen, Pajchel, et al., 2024).

At the same time, there is growing recognition that assessment, particularly formative assessment, has strong potential to support CT integration by making students' reasoning and problem-solving processes visible (Tang et al., 2020). Formative assessment evaluates current understanding and guides students' next steps (Black & Wiliam, 1998; Wiliam & Thompson, 2008). In mathematics, such assessments emphasize reasoning, representation, and justification, helping teachers interpret and respond to students' thinking (Björklund Boistrup, 2017; Kalinec-Craig, 2017). When CT is embedded in mathematics, formative assessment can reveal how students approach abstraction, decomposition, algorithms, and debugging, which are key dimensions of both computational and mathematical thinking. However, little empirical research has examined how teachers assess CT in mathematics classrooms or how their practices connect to students' learning (Bonner et al., 2021; Grover, 2017; Ye et al., 2023). This study addresses this gap by exploring how mathematics teachers assess CT in a classroom context. Drawing on classroom observations and interviews with two Norwegian teachers, this study investigates how teachers practice and reflect on assessment, with particular focus on formative assessment and feedback as mechanisms for supporting students' learning. The study aims to clarify how teachers' assessment practices shape students' engagement with computational and mathematical ideas and to deepen our understanding of how CT and mathematics can be meaningfully integrated through classroom practice. The guiding research question is: *How do teachers practice CT assessment in mathematics classrooms?*

In the following pages, we first turn to the relevant literature on CT, its relationship to school mathematics, and the challenges teachers encounter when integrating CT into mathematics teaching and assessment. Subsequently, we present a theoretical background of assessment, followed by a description of the methods used in this study. After presenting and discussing our findings, we conclude with practical and policy implications as well as suggestions for future research.

Literature review

What is Computational Thinking (CT)?

Papert (1980) introduced the term CT as part of his constructionist approach to education, emphasizing the importance of students' social and emotional involvement

in creating computational artifacts to explore concepts across disciplines. Papert saw CT not just as a tool for problem solving but as a way to concretize abstract ideas and empower learners to build meaningful models of the world (Lodi & Martini, 2021). Two decades later, Wing (2006) expanded the concept, framing it as a fundamental skill for everyone, not just computer scientists, by positioning it as a lens through which to understand the algorithmic fabric of modern life. Papert's focus on the creative and interdisciplinary applications of CT, combined with Wing's emphasis on CT as a problem-solving methodology and technical competency, highlights CT's relevance in the digital age (Lodi & Martini, 2021).

In an educational context, CT can today be understood as an approach to learning problem solving, with roots in computer science principles and applications that extend beyond programming or technology (Haseski et al., 2018; Papert, 1980; Shute et al., 2017). Although recent systematic reviews mapping the conceptual components of CT have revealed considerable diversity in interpretations among scholars, certain components have consistently emerged as core elements across reviews (see Grover & Pea, 2013; Hsu et al., 2018; Kalelioglu et al., 2016; Shute et al., 2017; Tikva & Tambouris, 2021; Zhang & Nouri, 2019). These core elements include abstraction (extracting the essence of a [complex] system; see Shute et al., 2017), algorithmic thinking (creating step-by-step instructions or procedures to solve a problem or perform a specific task systematically; see Doleck et al., 2017), decomposition (breaking down a complex problem into smaller, more manageable parts to understand and solve each component individually; see Kwon & Cheon, 2019), and debugging (identifying and fixing errors or flaws in a process, code, or system to ensure that it functions correctly; see Sun et al., 2024). Despite the definitional myriad (Haseski et al., 2018), a consensus can be traced in the literature, and many reviews draw on similar concepts.

In this paper, we base our understanding of CT on two frameworks, one by Shute et al. (2017) and the other by Barefoot Computing (2016), which is used in Norwegian education. Shute et al. (2017), based on a review, presented a definition comprising the following CT elements: decomposition, abstraction, algorithms, debugging, iteration, and generalization. In Norway, the curriculum follows the Barefoot Computing (2016) definition of CT, which consists of the following core concepts: logic (predicting and analyzing), algorithms (making steps and rules), decomposition (breaking down into parts), patterns (spotting and using similarities), abstraction (removing unnecessary detail), and evaluation (making judgment). It also includes CT approaches such as tinkering, creating, debugging, persevering, and collaborating (Barefoot Computing, 2016; NDET, 2019). Thus, although the Norwegian curricular definition is close to that of Shute et al. (2017), it includes more active CT approaches in addition to core concepts, making it slightly more process-oriented than many of the standard CT definitions.

Norway is an example of a country that integrates CT into its compulsory curricula.

In Europe, at least 24 countries have included it in their national curricula (Bocconi et al., 2022). How countries have chosen to integrate CT into their subjects varies, and while some integrate it as a separate subject (such as Cyprus, Greece, Poland, and the UK), other countries have decided to integrate it within other subjects, such as mathematics and technology (such as Finland, France, Sweden, and Norway) (Bocconi et al., 2022). A central pedagogical approach to teaching CT, both in subjects and as a separate subject, is programming-based instruction (Bocconi et al., 2022). However, securing qualified teachers is considered a challenge across Europe.

Empirical studies of CT in schools show that teachers often see connections between CT and mathematics, but that integrating CT into mathematics also poses challenges. In Norway, Nordby et al. (2022b) found that among primary school teachers new to CT, the Norwegian term "algorithmic thinking" led to misconceptions, with mathematics teachers interpreting CT as equivalent to learning algorithms for solving mathematical problems. Teachers were both sceptical about the introduction of programming and related concepts, but also positive towards including it as a necessary skill for the 21st century. Kaufmann and Stenseth (2020) found in a Norwegian study that when students engaged with programming in mathematics, they engaged in both code-focused argumentation, arguments based on mathematical considerations, and arguments based on generalization, showing that both programming and mathematics were present in students' argumentation. However, they found that a teacher with sufficient programming skills is required to facilitate students' mathematical argumentation when combining CT and mathematics.

Sands et al. (2018) studied how teachers in the US perceived CT and found that teachers inexperienced with CT saw it as connected to mathematics and related to logical thinking, displaying misconceptions similar to those of teachers in the study by Nordby et al. (2022b). Teachers in a study by Rich et al. (2019) similarly identified a relationship between CT and mathematics, drawing on their understanding of CT concepts and overlapping terminology between the disciplines. Teachers specifically linked CT to problem solving. Similar findings are reported in a study by Ukkonen, Yadav, et al. (2024), who found that CT terminology shapes how teachers perceive CT assessment, and that decomposition and debugging were easier to assess, whereas abstraction and patterns were more difficult to integrate into their subject teaching. Elementary teachers both in the US and Europe tend to focus on formative assessment of CT and see this as the most fruitful strategy for student assessment (Bocconi et al., 2022; Ukkonen, Pajchel, et al., 2024).

Collectively, the empirical studies show that teachers find CT integration possible when it is anchored in established subject traditions, particularly in mathematics, where connections to logical reasoning, problem solving, and programming are most tangible and teachable, but that misconceptions also arise, especially among teachers new to CT.

CT and teaching Mathematics

Different scientific disciplines, such as mathematics and science, connect to CT (Gadanidis et al., 2017). As Gadanidis et al. (2017, p. 77) explain, "there is a natural (and historical) connection between computational thinking and mathematics – in terms of logical structure and the ability to model mathematical relationships". From this perspective, CT is considered closely related to problem solving, and its components—such as problem decomposition, abstraction, algorithm design, debugging, iteration, and generalization—are also characteristics that are central to mathematics as a scientific discipline (Hansen & Hadjerrouit, 2022; Rich et al., 2019; Shute et al., 2017; Sneider et al., 2014). Considering the connections between CT and mathematics as scientific disciplines, a growing body of literature examines how these connections may support learning in both CT and mathematics (Bråting & Kilhamn, 2021; Hansen & Hadjerrouit, 2022; Kallia et al., 2021; Ye et al., 2023). Furthermore, the relationship between CT and programming concepts is not clearly bounded, and studies on CT assessment take different perspectives and disciplinary angles (Tikva & Tambouris, 2021). However, programming is frequently used as the primary approach to introducing CT in schools (Hsu et al., 2018), highlighting the close connection between CT and programming.

While the connection between CT and mathematics is evident from a disciplinary perspective, various scholars have identified the integration of CT into school mathematics as both challenging and fruitful. Challenges often relate to the less explicit link between CT and school mathematics. In a scoping review of studies published between 2006 and 2016, Hickmott et al. (2018) noted that while mathematics is often linked to CT, these connections are frequently incidental, with explicit links mainly observed in topics such as numbers, algebra, geometry, and statistics. The review highlighted the predominance of programming-focused studies, underscoring the need to explicitly develop knowledge about the relationship between mathematics and CT. Building on this, Nordby et al. (2022a) pointed out the lack of research exploring cases of mutual reinforcement of CT and mathematics—in which CT not only contextualizes but also drives the learning of new mathematical content.

Many opportunities for introducing CT in mathematics have been identified. For example, Ye et al. (2023) illustrated how CT concepts, such as loops and variables, align with mathematical practices, showcasing a reciprocal relationship between the two domains; however, challenges persist in translating concepts such as debugging into the mathematical context. Rich et al. (2022) observed that CT integration increases the cognitive demands of mathematical tasks, thereby fostering broader mathematical reasoning. Yadav et al. (2022) describe how CT shares many similarities with meta-cognition and that meta-cognition could provide a framework for making problem-solving and reasoning more explicit for students and teachers. Ng and Cui (2021) found that primary students engaged in CT could tackle advanced mathematical concepts, such as

algebraic expressions, without prior exposure, demonstrating CT's potential to scaffold complex ideas. However, challenges remain, as the semiotic and syntactic differences between CT and mathematics, particularly in concepts such as variables and functions, can both constrain and enable learning opportunities (Bråting & Kilhamn, 2021).

Teachers play a crucial role in integrating CT into school mathematics. Doing so effectively requires navigating distinct conceptualizations of shared mathematical and computational terms, such as variables, and establishing explicit connections between computational and mathematical thinking (Ng & Cui, 2021). Bråting and Kilhamn (2021) also highlighted the challenges of reconciling the divergent representational systems of CT and school mathematics, which can lead to conceptual mismatches if not addressed thoughtfully. Professional development and institutional support are essential to ensure that teachers are equipped to design CT-infused tasks that move beyond programming skills and emphasize applications in mathematical modeling and problem solving (Nordby et al., 2022b; Ukkonen, Pajchel, et al., 2024). However, the lack of frameworks or guidelines to support teachers' efforts in this domain underscores the need for further empirical research.

Theoretical background

Research suggests that formative assessment is crucial for supporting learning, not only by assessing current knowledge but also by providing direction for future learning (Black & Wiliam, 1998; Wiliam & Thompson, 2008). Key elements of formative assessment include clarifying and sharing learning intentions, classroom discussions, and tasks that reveal understanding; providing feedback that guides the learner forward; and activating students as resources for each other and as owners of their own learning (Wiliam & Thompson, 2008). It is essential for teachers to allow for self-reflection; as Boud (2008, p. 36) argued, "The most fundamental [aspect] is to create the circumstances that encourage students to see themselves as active agents in their own learning". Feedback can reduce the gap between current and desired understanding, and through feedback, students can see where they are in the learning process, where they are going, and how to get there (Hattie & Timperley, 2007). The feedback process enables students to see their progress and take ownership of their learning.

These principles take on a distinctive form in mathematics education, where assessment closely engages with students' reasoning, representation, and justification of ideas (Björklund Boistrup, 2017). In mathematics classrooms, formative assessment becomes a dialogue that helps teachers see how learners make sense of problems and construct meaning. Instead of focusing only on correctness, effective formative assessment highlights students' reasoning and treats confusion and errors as productive parts of learning (Kalinec-Craig, 2017). From this perspective, teachers can adapt feedback and

instruction to strengthen conceptual understanding (Palm et al., 2017). This approach aligns with broader efforts to assess complex mathematical competencies and equitable participation (Nortvedt & Buchholtz, 2018) and is increasingly supported by digital tools that provide more immediate, visual feedback (Drijvers & Sinclair, 2024).

Formative assessment in mathematics provides a broad perspective on CT assessment. It is a complex practice due to a multitude of approaches, context dependency, and the integration of CT with mathematics. Studying teachers' practices regards how teachers assess CT and mathematics, and whether they are assessed as integrated or separately. Assessment, especially formative assessment of CT within a subject, could provide feedback on learning related to both CT itself and the subject into which it is integrated.

However, CT assessment is not yet a well-established field (Bonner et al., 2021; Grover, 2017). In an overview of the (lack of) assessment in CT, Grover (2017), argues that there are few studies that have examined CT assessment. Tang et al. (2020) found that research on CT assessment focuses on computer science and robotics, while research focusing on STEM subjects lags behind. As Grover (2017) emphasized, there is a need to use broad assessment systems. There are concerns that students' use of computational constructs in code does not accurately reflect their CT abilities (Brennan & Resnick, 2012; Grover, 2017). To capture the different dimensions of CT, versatile forms of assessment should be prioritized, such as interviews, artifacts, and frequent quizzes, to elicit evidence of student understanding (Grover, 2017).

Only a few studies have explicitly examined the formative assessment of CT. Bonner et al. (2021) explored how students solved CT problems in combination with formative assessments, using self-regulated learning theory. They analyzed students' verbalizations ("think-alouds") and problem-solving processes and found that students who performed well used more iteration, task analysis, and planning, whereas frequent testing and debugging were associated with lower scoring in CT. Chevalier et al. (2022) studied feedback in CT and educational robotics, noting that delayed feedback was more effective than immediate feedback, leading to a less productive trial-and-error process. Hadad et al. (2020) studied informal formative assessment of CT in a maker environment and found that debugging facilitated peer- and mentor-led assessment. Grover (2017) emphasized the importance of assessing students' projects for deeper CT learning and found that students found these assessments more meaningful than summative tests. Similarly, in early childhood CT, Clarke-Midura et al. (2023) found that misconceptions can be reframed for productive formative assessments. These studies indicate that the state of the art regarding the formative assessment of CT shows how timing feedback plays a role and how planning, forethought, and metacognition can be important for learning CT. However, to our knowledge, there is scarce research on formative CT assessment in mathematics classrooms and on whether or how it can support learning mathematics and CT.

In this study, the conceptual focus is on assessment, particularly formative assessment as practiced by teachers, with an emphasis on feedback. The framework draws on Wiliam and Thompson's (2008) study, which describes formative assessment through key strategies such as clarifying and sharing learning intentions, using classroom discussions and tasks to elicit evidence of understanding, providing feedback that guides learners forward, and activating students as resources for one another and as owners of their own learning. Specifically, we examine how teachers assess CT and analyze the feedback they provide during mathematics teaching. As Hattie and Timperley (2007) noted, feedback enables students to understand where they are in the learning process, where they are heading, and how to move forward, making it important to consider how students respond to feedback. While feedback is the main focus, particular attention is given to how CT elements are reflected in it. In sum, this study approaches feedback as a form of formative assessment, focusing on how elements of CT are embedded within feedback practices.

Method

Context

The introduction of CT into the curricula of several countries has resulted in new requirements for teachers and students. In Norway, the context of this study, CT was integrated into mathematics, science, music, and arts in 2020 (NDET, 2019; Pajchel et al., 2024). Mathematics is the primary subject in which CT has been integrated, including basic algorithms in Grade 4, programming in Grade 5, and geometric explorations in Grade 6 (NDET, 2019). By Grade 10, students should be able to engage in data modeling, problem solving, and argumentation using programming.

The cases of Stephen and Lena

The work presented here was part of a larger design-based researcher–practitioner partnership project (McKenney & Reeves, 2018) about the integration of CT in mathematics and science. The collaboration lasted three years and involved workshops with discussions on the operationalization of CT and teaching approaches, classroom observations, and interviews with teachers, providing a realistic and rich context for this study. Case studies involve an in-depth analysis of a single, clearly defined unit (e.g., an individual, group, organization, or event) and are suitable for answering "why and how" questions (Yin, 2018, pp. 9-11). This article presents a qualitative case study that investigates two teachers' CT assessment practices, including an in-depth exploration of their formative assessment on CT-related mathematics tasks. Although this study is based on a focused dataset, it is interpreted in the context of broader data collection.

In line with Stake's (1995) guidelines, this case study is both intrinsic and instru-

mental. It focuses on two teachers who are actively engaged in CT assessment, aligning with the intrinsic case study approach, which seeks to understand a phenomenon by closely examining a specific case in its own right. Nonetheless, the study also aims to shed light on the broader phenomenon of CT assessment in mathematics education, consistent with the purpose of instrumental case studies, which use particular cases to gain insight into a larger issue. Although studying only two teachers limits generalizability, an in-depth examination of their practices provides rich examples of classroom situations and possible scenarios for implementing CT in mathematics. The teachers were purposefully selected as information-rich cases due to their interest and competence in programming and their direct experience with introducing CT. The aim was not statistical generalization but an in-depth exploration of classroom practices in their natural context, in line with the purposes of case studies (Stake, 1995; Yin, 2018). The small sample enabled intensive data collection, including observations, interviews, and artifacts, and allowed for fine-grained analysis of teaching and learning processes. In line with qualitative case study methodology (Yin, 2018), the findings contribute to an analytic understanding of assessing CT in mathematics and may inform those designing CT professional development for teachers.

The two teachers recruited through the researcher–practitioner partnership project are referred to by pseudonyms. The first is Stephen, who holds a master's degree in mathematics education, has studied informatics, and teaches Grades 7-10 (ages 12-15). His programming background is above average for Norwegian middle school teachers. The second teacher, Lena, has a four-year teacher education degree specializing in mathematics and science, and has completed professional development (PD) courses in CT and programming. She teaches Grades 1-7 (ages 6-12). Stephen and Lena were selected for their experience with integrating CT into mathematics and their ability to provide insight into expert teaching and assessment. Stephen had a developed portfolio of CT teaching materials before the study, while Lena actively applied strategies acquired through both PD and the project workshops. Stephen's lessons were observed over two years and Lena's over three, ensuring a thorough understanding of their teaching and teacher–student interactions. The other participants consisted of their students, who were selected in collaboration with the two teachers. A purposeful sampling technique with maximum variation was used (Suri, 2011) to select students; Stephen and Lena were asked to suggest students with disparate proficiency levels in mathematics and CT and who were expected to ask questions and engage in discussions. Pseudonyms are used in dialogue excerpts.

Lessons

The study follows lessons led by Lena and Stephen, all of which integrate mathematics and CT using Scratch, a block-based programming language widely used in Norwegian

schools. The lessons included teacher-led discussions followed by pair work. Both classes had prior Scratch experience.

Lena's Grade 6 class explored geometric figures during a 90-minute lesson. Figure 1 shows Lena drawing a triangle while providing feedback to a student. Students, working individually but seated in pairs, used Scratch's drawing tool to create rectangles, triangles, and polygons. The geometric figures drawn by the students gradually became more complex.

FIGURE 1

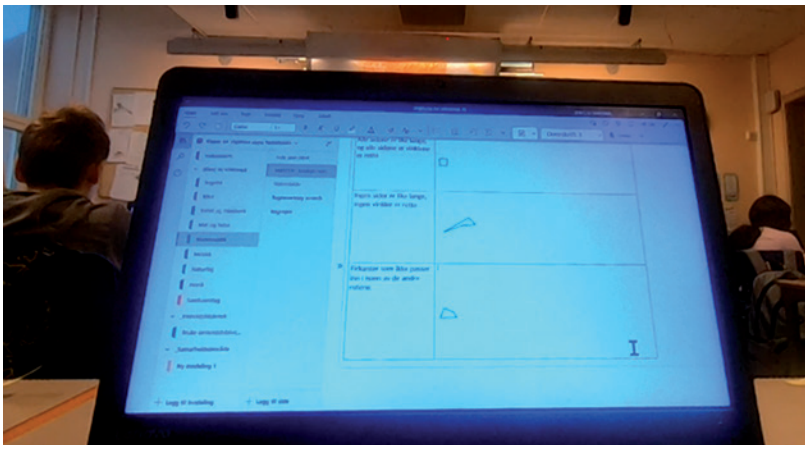
Lena providing feedback to a student and drawing a triangle

They documented their work with screenshots, which they later pasted into a worksheet (Figure 2). This lesson was followed by a teaching unit in which students programmed Bit:bot (a small, programmable robot car) to draw geometric figures on sheets of paper. This unit is not part of the core data but provides context for the analysis and interview with Lena.

Stephen's Grade 10 class involved two lessons. The first focused on equations, requiring students to implement a Fahrenheit-to-Celsius conversion in Scratch (Figure 3).

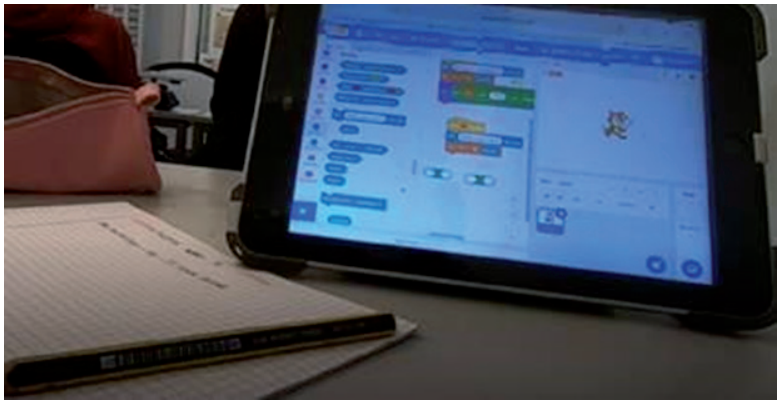
The second lesson introduced figurate numbers, sequences, and functions (Figure 4). Students created a Scratch program that generated an arithmetic sequence, progressing from recursion to a general function-based approach.

FIGURE 2



Lena's student's screen showing their worksheet

FIGURE 3



Stephen's student's screen showing their Scratch code and notebook

FIGURE 4

Graphical example of figurate numbers (Note: This example shows triangular numbers)

Data collection and analysis

To develop rich, multifaceted portraits of the two focal cases, Stephen and Lena, multiple forms of data were collected and analyzed in an integrated, iterative manner. Drawing on case study methodology (Creswell, 2013), data from individual semi-structured interviews and classroom observations were combined to explore how each teacher practices CT assessment in their mathematics classrooms. The individual interviews involved questions such as: "Can you describe a teaching unit involving computational thinking?", "How do you know whether a student has learned computational thinking?", and "What is needed for high/low goal achievement?". See Attachment A for an overview of the interview protocol for the individual interview. Classroom activities were captured using a multi-camera setup: One camera followed the teacher, while chest-mounted GoPro cameras recorded students' interactions with iPads, PCs, and notebooks. Observations were carried out by the first two authors and additional researchers, who also took detailed field notes.

Before data collection, the research project was approved by the Norwegian Agency for Shared Services in Education and Research (Sikt). An information letter was provided to teachers, students, and their guardians, explaining the aims of the project, what participation entailed, and that declining or withdrawing from the study would have no negative consequences. Informed consent was obtained from all participating teachers, students, and students' guardians. Students were also informed that they could withdraw at any time without consequences. All data was stored on an encrypted server accessible only to the researchers involved in the project and anonymized in dissemination. All procedures thus adhered to Norwegian ethical guidelines (Forskningsetikkloven, 2017; Norwegian Centre for Research Data, 2021).

Analytically, the interviews and classroom observations were not treated as discrete phases but as complementary and dialogic sources of insight. The initial round of analysis involved identifying occurrences of CT-related assessment both in teacher accounts and classroom interactions. Drawing on interaction analysis (Jordan & Henderson, 1995), we first examined the video recordings and selected classroom episodes involving teacher–student feedback.

Transcripts of both the interviews and video were then interpreted thematically (Braun & Clarke, 2019), allowing the teachers' reflections to inform the interpretation of classroom practice and vice versa. Following Saldaña's (2016) guidelines, the analytic process was largely exploratory and inductive, yet sensitized by conceptual frameworks. Notably, it was guided by the CT models created by Shute et al. (2017) and Barefoot Computing (2016), as well as by the existing literature on assessment understood as feedback, particularly the key strategies outlined by Wiliam and Thompson (2008). Through collaborative discussions within the research team, a broad set of codes related to CT- and mathematics-linked feedback was identified and is listed in Table I.

To strengthen analytical rigor, these codes were reviewed by the entire author team and refined through an iterative process inspired by the phases of thematic analysis (Braun & Clarke, 2006; Lincoln & Guba, 1985). The analysis was guided by the conceptual frameworks for CT and feedback mentioned above and was open to inductive codes through an exploratory approach. The team discussed various ways to group the codes, searching for teams that could combine and correlate the findings from the interviews (what teachers say) and the observations (what teachers do). Two key themes were generated across both cases: (1) the role of debugging in CT assessment-related productive failure and (2) the role of structured problem solving and mathematical reasoning in CT assessment, examining code features and quality, the latter understood as the use of appropriate programming constructs. These patterns, outlined in Table 1, frame the findings and discussion that follows.

TABLE 1

The analytic framework consists of the themes and categories used to analyze both the interviews and the video observation

Key themes	Categories	Explanation
The role of debugging in CT assessment-related productive failure	Debugging	Correcting the program to obtain the desired result
	Testing, tinkering	Running code to see the result
	productive failure	The teacher does not correct a code or procedure, leaving it to the students to analyze the result
	Pen-and-paper	Using drawings and calculations to understand the mathematical problem
	Persevering	Not giving up when facing problems
The role of structured problem solving and mathematical reasoning	Decomposition	Breaking down a problem or solution
	Formative checklist	A resource to structure problem solving
	Abstraction	Recognizing the essence of a problem
	Patterns, recursion	Recognizing recurring features of a problem or solution
	Algorithms	Stepwise instruction
	Quality of code	hard-coded solutions vs. more flexible programming concepts
	Variables and functions	Concepts and structures that occur both in mathematics and programming
	Reasoning	Express how one arrives at a result

TABLE 1

Feedback - relates to theme (I) debugging and (2) problem solving	Feed forward	Guiding the students to the next step(s) in a task
	Students as owners of their own learning	Supporting and engaging students in their learning process

FINDINGS

In this section, we present our findings on the cases of Stephen and Lena, each of which illustrates a different approach to CT practice. CT assessment practices are reflected in teachers' CT integration and feedback.

The case of Stephen

Stephen integrated CT with mathematics primarily through various Scratch projects. For example, in the interview, Stephen talked about an earlier project in which students were tasked with programming a multiplication game. In his assessment, Stephen sought to link key CT elements with mathematics but found assessing CT challenging: "[I] still feel like I don't quite have a hundred percent grasp of it [CT], but I do have my own interpretation". His interpretation emphasized programming, problem decomposition, and problem solving, viewing CT in mathematics as the ability to break down a problem or situation into smaller, more manageable subproblems. This structured problem-solving method was the foundation of his approach to connecting CT with mathematics.

To facilitate the students' progress, Stephen provided a table that decomposed problems into parts. For example, the multiplication game could be divided into subproblems, starting with programming an input request for an integer, followed by tasks such as determining the number of rounds and providing feedback to the user. He also provided a checklist that students could use to indicate whether and how they solved each subtask. As he explained in the interview, "The students have to create such flowcharts in addition to writing code, and then they have to break up the problem, which is 'How do you create a multiplication game?', into individual subprograms". This checklist served as a formative assessment tool, helping him structure the decomposition process for students. Later, he used it as a basis for discussing the results with his students. While he found this beneficial for students who struggled, he acknowledged that it limited their ability to independently decompose problems.

Stephen's assessment process involved evaluating students' progress using formative checklists and final projects through summative end-of-semester discussions. How-

ever, he was still searching for effective ways to assess CT in mathematics. While the checklist allowed students to reflect on their work and understand task objectives, formative assessment components such as planning, forethought, and peer assessment received less emphasis. Thus, although the checklist helped him share and clarify learning intentions, it limited students' ownership of their learning. Notably, Stephen did not mention in-classroom observation or feedback as part of his assessment practices.

Stephen's approach focused primarily on problem decomposition, with less emphasis on other CT elements or mathematical content. His approach to assessing subproblems included checking whether students had hard-coded solutions or programmed functions, indirectly assessing abstraction without explicitly referencing the official CT definition (NDET, 2019). He expressed that integrating CT into mathematics was challenging: "I sometimes feel that I run out of ideas, like in which context I can apply the programming logic and the mathematics so that it is [not only] interesting but also challenging—but not too challenging". He also stated that his concerns about effectively assessing CT in mathematics were connected to the integration of CT into mathematics. In particular, he found the scarcity of tasks that meaningfully integrate both domains challenging.

Stephen's in-classroom formative assessment was characterized by two main approaches: feedback on debugging and feedback on variables and functions. Feedback on debugging corresponds to the first identified theme, the role of debugging in CT assessment related to examining code features and quality, while feedback on variables and functions connects both to debugging and to the second theme about the role of structured problem solving and mathematical reasoning in CT (outlined in section 'Data collection and analysis').

Regarding debugging, students showed reluctance to debug their own programs, often seeking feedback from the teacher instead. One of their tasks was to implement a formula for temperature conversion. Working on this task, two students, Mia and Amalie, wanted reassurance that their programs were correct before running them. After completing a Celsius-to-Fahrenheit conversion program, they spoke with the teacher instead of testing their code:

Stephen: Did you get it, Mia?

Mia: I think so.

Stephen: Yes! Let's see what you have over there... It looks very nice, but have you tried to test it to see if it works?

Mia: No.

Stephen: Maybe you should try it? Try to check both this part and that part. Then try to test, for example, C to F first, and then give it an integer, which is easy to check. And then double-check by calculating by hand if you are unsure.

Despite this guidance, Mia did not run her program and instead started doodling in her notebook. A similar exchange occurred with Amalie, who also did not test her program until prompted to. Once she did, Stephen encouraged further testing: "Then try C to F, for example, and then one more [temperature]". Testing and debugging a program is a relatively complex task, and students may need help to break it down into smaller, systematic steps. These examples show how Stephen offers feedback that guides the students forward by suggesting concrete procedures for debugging.

While the teacher's questions encouraged them to test their programs, the teacher did not explicitly address whether the students understood what the expected output should be, limiting the prediction and analysis, which are parts of CT. Debugging can be difficult without a clear expectation of the result and thus a shared understanding of the learning goals. While some elements of forethought were present, Stephen's feedback guiding the next step focused primarily on immediate debugging.

A connection between problem solving and CT was evident in Stephen's feedback regarding variables and functions. He needed to provide explicit feedback on variables and functions, aiming to help students with variables, particularly in translating mathematical functions into functions in Scratch. Amalie was confident that the variable should be included in the program, but sought clarification from Stephen on how to implement it via a function in Scratch. He guided her with feedback on how to express fractions and multiplication in code. The other student, Mia, however, needed feedback about variables:

Mia: Yes, but I don't remember how to create a variable.

Stephen: You have C here, right? [...]

Mia: But I must have "set C to answer."

Stephen: Yes, but don't think about it quite yet. Try to... The only thing you need to do is, if you see the expression we have there, you're just going to do the same expression for that formula. So, the task is to create that calculation here, just as a completely separate, isolated part, with the operators and such.

Mia struggled to define a variable and expressed frustration with fractions and numbers. Stephen tried to aid her with feedback on how to express the variable. However, the feedback failed to help her navigate variables, a concept at the intersection of programming and mathematics. Later, when working with the recursive function for figurative numbers using the formula, the same student, Mia, encountered errors. To guide the debugging process forward, Stephen advised her to write the formula in her notebook:

Stephen: [...] If you take the notebook...

Mia: Yes.

Stephen: ...and then you write that formula here literally, then you will quickly see where the error is because the formula should really be $6n + 1$, right?

Mia: Yes.
Stephen: So, what is actually written here? [Points to Mia's code block.] Is it 6*n* + 1, or is it something else?

In her notebook, Mia wrote "6·figure + 1" instead of "6*n* + 1", indicating confusion between variables in figurate numbers and the variable in programming. In the next task, she miswrote the expression as "8 + 4" instead of "8*n* + 4", omitting the variable entirely. The teacher's feedback to use pencil and paper to break down the problem helped debug the program and revealed confusion between the mathematical concept of a variable and a variable in programming, understood as a pointer to some value.

Overall, Stephen's case highlights two key aspects of CT instruction and feedback in mathematics. First, his feedback related to debugging, a fundamental CT skill. Testing and making the program work properly is a central skill in systematic problem solving, which, in Stephen's view and practice, is the basis for the integration of CT and mathematics. The classroom observation shows how the mathematics content sets the parameters for testing the program and how the teacher guides the students forward. An identified opportunity was that students could benefit from feedback that extends beyond immediate corrections to include planning and procedures that could be applied in future tasks. Second, for Stephen, a challenging aspect of CT assessment is the integration of CT into mathematics through meaningful applications. Students' need for feedback, particularly regarding variables and functions, shows both how they develop an understanding of related mathematics and programming concepts and how code features may play a role. While feedback that activated students as owners of their own learning (e.g., using familiar tools such as pencils and paper) helped bridge the gap between CT and mathematics, providing feedback that would have helped the students to succeed in the task proved difficult. If students are provided feedback that includes the intended output of a program and expected results, they may obtain direct feedback from the program by systematically running and testing their solutions, but in Stephen's students' case, this was still a future step.

The case of Lena

Lena's approach to CT assessment primarily focused on gaining insights into students' reasoning. She acknowledged in the interview that her assessment methods were not always consciously planned, but emphasized the importance of understanding how students think. She typically engaged with students by asking questions such as "What do you have in mind here?", "Why have you chosen this block?", "How did you do it?", and "If it doesn't work, what should you have done differently?". Lena perceived CT assessment as challenging because there is "so much hidden thinking". Therefore, she

practices CT assessment, "much like most school assessments, as primarily formative," focusing on providing feedback.

According to Lena, programming- and CT-related tasks are often completed in groups and may lead to an uneven distribution of effort among students. Therefore, she sometimes found it challenging to gauge individual performance. To address this, she considered written logs as an alternative approach to individual assessment. In this approach, students would describe their thought processes and what they have done so that Lena could gauge their understanding.

The topic of the studied teaching unit was geometric figures and patterns. Lena used mathematical learning goals in assessment, evaluating whether students "have achieved what they set out to do, if they have managed to create a pattern". Her aim was to integrate formative assessment into her teaching by supervising students through the task, ensuring that they "get on the right track with the codes" and providing feedback on their programming process. After the teaching unit that constitutes the core data for Lena's case, students worked on a programming task in which they used Bit:bot to draw patterns on large sheets of paper. Lena expressed in the interview a desire to develop this approach, letting the students consider the patterns as a kind of artwork. Lena saw that the resulting drawings could be a "product at the end that could be a summative assessment of everything in this task". Lena considered geometric figures, which were part of the mathematics content, central to both formative and summative assessment.

Regarding the criteria for high and low achievement, Lena pointed to features of the code. She explained that higher achievement could be indicated by the simplicity of the code, as simple code—understood as efficient code—demonstrates an understanding of algorithms and abstraction. An example Lena gave was "using loops to repeat instead of entering the same block ten times". However, she also believed that a functioning code, one that accomplishes its intended task, is already an achievement. In the drawing task given to the students during the teaching unit of this study (Figure 2), completing a geometric figure was a central learning goal. In both the interview and the classroom, Lena indicated that qualities such as algorithm simplicity and efficiency may serve as assessment criteria.

Lena's formative assessment, as observed in the classroom, is characterized by two main approaches: a) reasoning and productive failure, and b) debugging as an approach to expressing mathematical thinking. Debugging corresponds to the first overarching theme mentioned in section 'Data collection and analysis' (the role of debugging in CT assessment related to examining code features and quality), while reasoning and productive failure correspond to the second overarching theme (the role of structured problem solving and mathematical reasoning in CT assessment). The dialogue below illustrates how Lena guided students through reasoning about code elements representing straight lines and turns by progressively guiding them forward.

Lena: Yes, so now you get to draw a straight line. And now the question is, how can we draw a square? What does it take?
Andy: Up and down and so on. [Draws a figure in the air]
Lena: Yes! Are there any blocks here we can use then? Just look at the ones you've got up already. [Points to the movement blocks]
Andy: Turn round?
Lena: For example…
Andy: 15 degrees?
Lena: A tip could be to use more steps, put a higher number of steps... or also use this [points to a turn block], but go number [of steps], write 50, for example, because then you get a longer line each time.
Andy: Oh yes...
Lena: And if you try now... [points to flags, the Scratch-animal moves and turns 15 degrees each time]

Lena's emphasis on reasoning and productive failure included elements of persevering with the task. Persevering is one of the CT elements in the Norwegian curriculum. When the student initially suggested turning 15 degrees to draw a square, Lena did not immediately correct him. Instead, she encouraged experimentation, which aligns with her approach to exploratory learning, thereby encouraging students to stay with the task and emphasizing the importance of perseverance.

As students progressed in developing drawing algorithms, Lena introduced the concept of loops as an aspect of algorithms and abstraction. Once students successfully coded a basic line and turn, they were guided to complete a square by executing the code multiple times in a loop. In the excerpt below, Lena supervises a student who is about to draw a triangle. Lena reuses the code for drawing a square that Betsy has already mastered to introduce the concept of loops. Betsy then takes it a step forward by modifying it to obtain a triangle.

Lena: What might be a good thing to put in? If we go to what's called "control", if you go back to it... If you look at control, then you see that there is something called repeat. So instead of having to press it [the run button] many times, if you put it sort of around there... yeah, like that. And then repeat four times—then if it's going to be a square.
Betsy: Three now?
Lena: So, if we take three, repeat three times for a triangle.

The use of loops simplified and automated the code, but Lena framed the rationale in terms of geometry rather than programming principles. This approach helped students transition naturally from drawing squares to drawing triangles and other geometric figures, reinforcing their skills in abstraction and patterns.

A frequently observed type of feedback involved debugging, particularly when students worked on drawing triangles, which were more challenging than squares. The instruction for the sprite (an object or character in Scratch that you can program to perform actions in a project, such as moving, talking, or changing appearance) that draws lines is how many degrees it should turn with respect to the direction it is pointing, while in typical mathematics instruction, one focuses on the angles inside a geometric figure. Lena guided students in setting the turning angle by linking their programming errors to mathematical reasoning using pen and paper. When students incorrectly inserted 60 degrees as the turning angle, Lena prompted them as follows:

> *Lena*: Hmm... and then it's important—how many degrees? And all the angles... the sum of angles... [Writes on the sheet: "sum of angles"] It's 180 degrees, but that's all the angles together. And how many angles are there? One, two, three? [Points to the drawing of the triangle] So then I can think 180 divided by the three angles. Then you can think, first 18 divided by three, that is?
>
> *Claire*: Six.
>
> *Lena*: Hmm... and then I have to add that zero. So, every angle here has to be 60 degrees. So, if you try it first... hmm... and then you have to put in one of those... if you take away the whole repeat 4 there... move it to the side... and then you press the flag and see what happens.

Once the student observes the error—that the angle the sprite turned is different from what was expected, thereby not forming a triangle—Lena further prompts the student:

> *Lena*: And do you see? It doesn't turn out quite as we thought. And that's because it turns 60 degrees in a different direction than we do. So, we have to think, okay, 60—what do I have to add to 60 to get 180? What is the opposite angle? Think 180 minus 60.
>
> *Claire*: 120.
>
> *Lena*: Then it turns the other way. So, try 120 instead, and you'll get a triangle.

By linking debugging to mathematical concepts, Lena reinforced productive failure, allowing students to test, analyze, and adjust their code independently. Thus, we see how the themes of debugging and reasoning also overlap, illustrating the iterative nature of the CT process and how CT elements intertwine with mathematics.

Overall, the observations highlight Lena's feedback strategies, which are characterized by guidance that moves the learner forward. The feedback emphasized algorithms and code abstraction, which are closely related to decomposition and debugging. Her approach aligned with her belief that understanding mathematical figures and patterns can be expressed in terms of code and the use of appropriate constructs (e.g., loops),

which are related to abstraction and algorithms, and that such understanding constitutes evidence of CT. Her feedback guided students forward through exploratory learning processes, allowing productive failure, encouraging perseverance with the task, and linking programming to mathematical reasoning. These findings offer insights into how Lena uses feedback to cultivate both computational and mathematical thinking in her students.

Discussion

Although interest in CT assessment is growing in research internationally, the field remains in its early stages (Bocconi et al., 2022; Tang et al., 2020). Few researchers have examined how teachers navigate this complex process. This study highlights how two teachers, Stephen and Lena, approached the assessment of CT in mathematics through formative assessment. Their cases reveal different assessment practices, showcasing how CT assessment is intertwined with how CT is integrated into mathematics classroom practices.

The relationship between CT and Mathematics

Our findings provide empirical support for the close relationship between CT and mathematics that has been theorized in previous research. From a disciplinary perspective, both domains share core practices such as abstraction, decomposition, and algorithmic reasoning (Gadanidis et al., 2017; Kallia et al., 2021; Shute et al., 2017). In the cases of Stephen and Lena, these shared practices were not treated as separate competencies but emerged through assessment as intertwined forms of sense-making. Teachers' feedback consistently connected computational constructs to mathematical meaning, for example, when debugging was framed through numerical verification or geometric reasoning, or when abstraction was assessed through the use of variables, functions, or loops that represented mathematical generalizations. Such findings align with prior research suggesting that CT can function as a mediating practice for mathematical reasoning rather than as an external or add-on skill (Bråting & Kilhamn, 2021; Ye et al., 2023). Bråting and Kilhamn (2021) analyzed a task involving programming a robot's path and raised questions about whether differences in mathematics and programming (for example, differences in the semiotic representations, such as functions) may blur mathematical concepts. In Lena's case, her feedback links the movement of the sprite to geometrical figures and the introduction of loops and variables, which, through abstractions of angles, lead to more general drawing algorithms. Likewise, Stephens's emphasis on decomposition, variables, and functions leads students toward more general solutions through abstraction. Thus, the teachers' approaches showed how mathematical content was displayed through CT. At the same time, the difficulties students

experienced, particularly when translating between mathematical and computational representations, reflect concerns raised in the literature about semiotic and conceptual mismatches between programming and school mathematics (Bråting & Kilhamn, 2021; Hansen & Hadjerrouit, 2022). Importantly, our findings extend this body of work by demonstrating that formative assessment and feedback are key sites where these connections are negotiated in classroom practice, an issue we discuss in more detail in Section 'Bridging CT and Mathematics in assessment'.

Divergent approaches to CT assessment

The two teachers had different approaches to assessing CT within mathematics. Stephen's approach was structured around problem decomposition, in which students were expected to break down mathematical problems into computationally solvable subproblems. However, he was uncertain about how to make the connections between CT concepts and mathematics explicit, reflecting broader challenges in defining CT assessment criteria. Stephen's assessment relied on structured tools such as checklists. His decomposition approach resembles structured problem solving, which is a central element connecting CT and mathematics (Kallia et al., 2021). At the same time, observations of students' testing and debugging revealed a need for providing concrete steps, as students tended to depend on teacher validation instead of developing their own independent problem-solving strategies. In sum, Stephen's feedback approach concentrated on debugging, structured decomposition, and students' programmes. Similar to Ye et al.'s (2023) findings, he encouraged students to apply mathematics to anticipate and interpret outputs from their CT-artefacts.

By contrast, Lena focused on abstraction and reasoning, integrating geometric representations to foster connections between CT and mathematics, an approach identified by Ye et al. (2023) as fruitful for learning both. Visualization is a frequently used approach to integrating CT into mathematics (Kallia et al., 2021), and Lena assessed students' CT processes through productive failure, emphasizing explanation and justification in addition to emphasizing functional correctness. She also used students' knowledge of geometrical figures in debugging and development of the drawing algorithms. By encouraging students to articulate their reasoning, this approach may support both computational and mathematical understanding. Ye et al. (2023) highlight that applying mathematics to the construction of CT artifacts may facilitate meaningful CT-based mathematics learning.

Although their approaches differed, both teachers integrated CT into mathematics by having students apply mathematical knowledge when working with CT artifacts and emphasizing different aspects of CT in their feedback and assessments.

Debugging as a key CT assessment practice

A recurring theme in both classrooms was the use of debugging as an assessment tool. Debugging, testing, and incremental modifications are integral to CT (Rich & Strickland, 2020) and featured prominently in both Lena's and Stephen's feedback. In Stephen's classroom, debugging was explicitly linked to mathematical verification, with students being encouraged to manually check computations before relying on program outputs. However, students often sought teacher confirmation before running their code instead of engaging in independent testing or peer review. This dependency reflects challenges identified in prior research (Ukkonen, Pajchel et al., 2024), indicating that, compared to traditional mathematics tasks with clear right-or-wrong answers, students are not accustomed to the open-ended nature of programming. Similarly, Bråting and Kilhamn (2021) argue that CT is more concerned with the process than the result, and, as our findings show, this also represents a challenging shift in teachers' assessment practices towards scaffolding and providing feedback on problem-solving. Booth teachers in our study acknowledge the need for supporting students' reasoning. However, in line with Stephen's reflection, how much help students should get, and in what form, is a justified question. Rich et al. (2022) find that integrating CT in mathematics tasks may impose higher cognitive demands but also provide students with practices that make them accessible. Such CT-based procedures may be transferred to other tasks as they, in a broader sense, represent meta-cognitive strategies and can make students' thinking processes more visible (Yadav et al., 2022).

Without a deliberate focus on testing and debugging strategies, students often find them challenging (Hansen & Hadjerrouit, 2022). Similarly, our findings indicate a need for explicit instruction and feedback on these strategies. Lena fostered a culture of iterative debugging and self-reflection, encouraging students to experiment with code even when errors were expected. This aligns with Ye et al.'s (2023) findings, which highlight debugging as a means of promoting high-level problem solving and self-regulated learning. Moreover, they emphasize that anticipating, interpreting, and making sense of outputs are central to mathematical reasoning and inquiry in CT tasks. Programming can be seen as a process of inquiry in itself, and targeted testing of changes in code can be framed as an "information seeking activity" (Litherland & Kluge, 2023, p. 512) as opposed to random trials. Debugging, from this viewpoint, can serve as a mediating process for mathematical reasoning—paralleling both Stephen's and Lena's pedagogical approaches. This view is supported by Chevalier et al. (2022), who caution that overreliance on rapid trial-and-error approaches can hinder deeper learning if not balanced with structured reflection. Our observations suggest that while feedback on debugging is needed for students' CT development, it should be coupled with explicit strategic debugging instruction to ensure that students move beyond surface-level troubleshooting.

Bridging CT and Mathematics in assessment

A significant finding of this study is that CT feedback and assessment in mathematics are deeply intertwined with the integration of CT and mathematics. The nature of CT assessment shifts from standard programming assessment when situated within mathematical practice.

Despite the conceptual overlap between CT and mathematics, students often struggle to bridge the two disciplines (Hansen & Hadjerrouit, 2022). Both teachers attempted to address this gap by designing tasks that explicitly linked mathematical reasoning to computational representations, aligning with meaningful approaches highlighted by Bråting and Kilhamn (2021) and Ye et al. (2023). Feedback strategies encouraged students to use traditional mathematical tools (pen-and-paper calculations) alongside programming tasks, helping them translate mathematical expressions into computational logic. Although mathematical knowledge can support the creation of CT artifacts, differences in representations present a challenge (Ye et al., 2023). In Stephen's class, this became apparent when studying his feedback, which showed how differences in representational systems between mathematics and programming created learning challenges. Students struggled to conceptualize mathematical functions and variables within the Scratch environment, highlighting a common issue in CT integration (Bråting & Kilhamn, 2021). Lena's approach to CT and mathematics integration was more intuitive; however, students were still confused and needed feedback as they initially translated geometric drawings into programmed sprite movements before refining their work using iterations and loops. The feedback practices of both teachers mirror those mentioned in the findings of Clarke-Midura et al. (2023), who advocated using formative assessment to leverage students' misconceptions as productive learning opportunities.

The two case studies illustrate, through the lens of assessment, the distinct challenges associated with integrating CT into mathematics, particularly when using geometry and functions. These findings underscore the importance of examining CT-mathematics assessment within the context of diverse mathematical topics, highlighting that a uniform approach is inadequate. Therefore, tailored assessments are needed in CT-mathematics instruction.

Limitations

As with any empirical research, this study has certain limitations that affect its implications (Maxwell, 2013). The study is grounded in a qualitative case study design, which prioritizes depth and contextual understanding over breadth (Stake, 1995; Yin, 2018). The insights presented are drawn from two teachers working in specific school settings in Norway and thus reflect their particular experiences, interpretations, and classroom practices related to CT assessment in mathematics, thereby limiting generalizability

(Lincoln & Guba, 1985). The participating teachers were both relatively experienced and positively inclined toward integrating CT into their teaching. Because teachers working under different constraints or with different levels of CT familiarity may have approached assessment differently, further research across diverse contexts is needed. Finally, while the findings are not generalizable in a statistical sense, they may offer transferable insights that educators and researchers working in similar contexts can relate to and reflect upon (Tracy, 2010). These limitations are characteristic of qualitative case study research and highlight both the situated nature of the findings and the potential for future studies to expand on the work presented here.

Implications and recommendations

Our findings suggest the need for further research on the interplay between CT and mathematics in assessment. Special attention should be given to what the literature describes as overlapping terminology and conceptual connections (Bråting & Kilhamn, 2021) to reduce potential confusion among students. One crucial area of focus is the role of debugging strategies in assessment. While debugging is a valuable tool for evaluating problem-solving skills (Fitzgerald et al., 2008; Sun et al., 2024), students should be encouraged to engage in strategic debugging rather than rely solely on trial-and-error methods. This can be achieved through structured self-evaluation techniques (Lee et al., 2023; Lishinski & Yadav, 2021; Litherland & Kluge, 2023) and peer feedback mechanisms (Fang et al., 2022), which promote a more reflective and analytical approach to problem solving. As systematic CT-based problem-solving overlaps with metacognitive processes (Yadav et al., 2022), it may be beneficial for broader learning. Additionally, explicit connections between CT and mathematics could be emphasized in the design of tasks and the assessment thereof. Rich et al. (2022) emphasize that CT-integration through procedures should connect to big mathematical ideas and thus provide teachers with insights into students' reasoning. Teachers should ensure that mathematical reasoning is linked to CT principles, helping students recognize the computational underpinnings of mathematical concepts and fostering a deeper understanding of both disciplines.

A balanced approach to assessment is also essential. Targeted feedback through questions, as seen in Lena's class, encourages reasoning and exploration, whereas structured assessment tools, such as those used by Stephen, provide clarity and consistency. Combining these approaches, that is, leveraging the strengths of both open-ended inquiry and scaffolded assessment, could contribute to student learning. Finally, further research is needed to explore how CT both enables and constrains mathematical learning, particularly related to assessment practices. Investigating different age groups and learning contexts could offer deeper insights into effective strategies for integrating CT and mathematics in a way that enhances students' conceptual understanding and problem-solving abilities. Eventually, the research field would also benefit from the devel-

opment of pedagogical feedback and assessment frameworks, for example, through a grounded theory approach (Strauss & Corbin, 1994). Clearly defining the intersection between CT and mathematics could help standardize assessment approaches, ensuring consistency and coherence in how these concepts are assessed.

References

Barefoot Computing. (2016). *Computational thinking concepts and approaches*. https://www.barefootcomputing.org/

Björklund Boistrup, L. (2017). Assessment in mathematics education: A gatekeeping dispositive. In H. Straehler-Pohl, N. Bohlmann & A. Pais (Eds), *The disorder of mathematics education* (pp. 209-226). Springer. https://doi.org/10.1007/978-3-319-34006-7_13

Black, P., & Wiliam, D. (1998). Assessment and classroom learning. *Assessment in Education: Principles, Policy & Practice*, *5*(1), 7-74. https://doi.org/10.1080/0969595980050102

Bocconi, S., Chioccariello, A., Kampylis, P., Dagienė, V., Wastiau, P., Engelhardt, K., Earp, J., Horvath, M. A., Jasutė, E., Malagoli, C., Masiulionytė-Dagienė, V., & Stupurienė, G. (2022). *Reviewing computational thinking in compulsory education* (JRC128347). Publications Office of the European Union.

Bonner, S., Chen, P., Jones, K., & Milonovich, B. (2021). Formative assessment of computational thinking: Cognitive and metacognitive processes. *Applied Measurement in Education*, *34*(1), 27-45. https://doi.org/10.1080/08957347.2020.1835912

Boud, D. (2008). How can practice reshape assessment? In G. Joughin (Ed.), *Assessment, learning and judgement in higher education* (pp. 31-44). Springer. https://doi.org/10.1007/978-1-4020-8905-3

Bråting, K., & Kilhamn, C. (2021). Exploring the intersection of algebraic and computational thinking. *Mathematical Thinking and Learning*, *23*(2), 170-185. https://doi.org/10.1080/10986065.2020.1779012

Braun, V., & Clarke, V. (2019). Reflecting on reflexive thematic analysis. *Qualitative Research in Sport, Exercise and Health*, *11*(4), 589-597. https://doi.org/10.1080/2159676X.2019.1628806

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 Annual Meeting of the American Educational Research Association,* Vol. 1, Vancouver, 13-17 April 2012. http://scratched.gse.harvard.edu/ct/files/AERA2012.pdf

Chevalier, M., Giang, C., El-Hamamsy, L., Bonnet, E., Papaspyros, V., Pellet, J.-P., Audrin, C., Romero, M., Baumberger, B., & Mondada, F. (2022). The role of feedback and guidance as intervention methods to foster computational thinking in educational robotics learning activities for primary school. *Computers & Education*, *180*, 104431. https://doi.org/10.1016/j.compedu.2022.104431

Clarke-Midura, J., Lee, V. R., Shumway, J. F., Silvis, D., Kozlowski, J. S., & Peterson, R. (2023). Designing formative assessments of early childhood computational thinking. *Early Childhood Research Quarterly*, *65*, 68-80. https://doi.org/10.1016/j.ecresq.2023.05.013

Creswell, J. W. (2013). *Qualitative inquiry & research design: Choosing among five approaches* (3rd ed.). Sage.

Doleck, T., Bazelais, P., Lemay, D., Saxena, A., & Basnet, R. (2017). Algorithmic thinking, cooperativity,

creativity, critical thinking, and problem solving: Exploring the relationship between computational thinking skills and academic performance. *Journal of Computers in Education*, *4*(4), 355-369. https://doi.org/10.1007/s40692-017-0090-9

Drijvers, P., & Sinclair, N. (2024). The role of digital technologies in mathematics education: Purposes and perspectives. *ZDM Mathematics Education, 56*(2), 239-248. https://doi.org/10.1007/s11858-023-01535-x

Fang, J. W., Shao, D., Hwang, G. J., & Chang, S. C. (2022). From critique to computational thinking: A peer-assessment-supported problem identification, flow definition, coding, and testing approach for computer programming instruction. *Journal of Educational Computing Research*, *60*(5), 1301-1324. https://doi.org/10.1177/07356331211060470

Fitzgerald, S., Lewandowski, G., McCauley, R., Murphy, L., Simon, B., Thomas, L., & Zander, C. (2008). Debugging: Finding, fixing and flailing, a multi-institutional study of novice debuggers. *Computer Science Education*, *18*(2), 93-116. https://doi.org/10.1080/08993400802114508

Forskningsetikkloven. (2017). *Lov om organisering av forskningsetisk arbeid* (LOV-2017-04-28-23) [Law about the organization of research ethics]. Lovdata. https://lovdata.no/lov/2017-04-28-23

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

Grover, S. (2017). Assessing algorithmic and computational thinking in K-12: Lessons from a middle school classroom. In P. J. Rich & C. B. Hodges (Eds), *Emerging Research, Practice, and Policy on Computational Thinking* (pp. 269-290). Springer.

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

Hadad, R., Thomas, K., Kachovska, M., & Yin, Y. (2020). Practicing formative assessment for computational thinking in making environments. *Journal of Science Education and Technology*, *29*(1), 162-173. https://doi.org/10.1007/s10956-019-09796-6

Hansen, N. K., & Hadjerrouit, S. (2022). Analyzing students' computational thinking and programming skills for mathematical problem solving. In D. Ifenthaler, D. G. Sampson & P. Isaías (Eds), *Open and inclusive educational practice in the digital world* (pp. 155-173). Springer Nature. https://doi.org/10.1007/978-3-031-18512-0_10

Haseski, H. I., Ilic, U., & Tugtekin, U. (2018). Defining a new 21st century skill-computational thinking: Concepts and trends. *International Education Studies*, *11*(4), 29. https://doi.org/10.5539/ies.v11n4p29

Hattie, J., & Timperley, H. (2007). The power of feedback. *Review of Educational Research*, *77*(1), 81-112. https://doi.org/10.3102/003465430298487

Hickmott, D., Prieto-Rodriguez, E., & Holmes, K. (2018). A scoping review of studies on computational thinking in K-12 mathematics classrooms. *Digital Experiences in Mathematics Education*, *4*(1), 48-69. https://doi.org/10.1007/s40751-017-0038-8

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

Jordan, B., & Henderson, A. (1995). Interaction analysis: Foundations and practice. *The Journal of the Learning Sciences*, *4*(1), 39-103. https://doi.org/10.1207/s15327809jls0401_2

Kalelioglu, F., Gülbahar, Y., & Kukul, V. (2016). A framework for computational thinking based on a systematic research review. *Baltic Journal of Modern Computing*, *4*(3), 583-583.

Kalinec-Craig, C. A. (2017). The rights of the learner: A framework for promoting equity through formative assessment in mathematics education. *Democracy & Education, 25*(2), 1-9. http://democracyeducationjournal.org/home/vol25/iss2/5

Kallia, M., van Borkulo, S. P., Drijvers, P., Barendsen, E., & Tolboom, J. (2021). Characterising computational thinking in mathematics education: A literature-informed Delphi study. *Research in Mathematics Education*, *23*(2), 159-187. https://doi.org/10.1080/14794802.2020.1852104

Kaufmann, O. T., & Stenseth, B. (2020). Programming in mathematics education. *International Journal of Mathematical Education in Science and Technology*, *52*(7), 1029-1048. https://doi.org/10.1080/0020739X.2020.1736349

Kwon, K., & Cheon, J. (2019). Exploring problem decomposition and program development through block-based programs. *International Journal of Computer Science Education in Schools*, *3*(1), 3-16.

Lee, S. W. Y., Tu, H. Y., Chen, G. L., & Lin, H. M. (2023). Exploring the multifaceted roles of mathematics learning in predicting students' computational thinking competency. *International Journal of STEM Education*, *10*(1), Article 64. https://doi.org/10.1186/s40594-023-00455-2

Lincoln, Y. S., & Guba, E. G. (1985). *Naturalistic inquiry*. Sage.

Lishinski, A., & Yadav, A. (2021). Self-evaluation interventions: Impact on self-efficacy and performance in introductory programming. *ACM Transactions on Computing Education*, *21*(3), 1-28. https://doi.org/10.1145/3447378

Litherland, K., & Kluge, A. (2024). Learning to program as empirical inquiry: Using a conversation perspective to explore student programming processes. *Computer Science Education*, *34*(3), 495-519. https://doi.org/10.1080/08993408.2023.2290410

Lodi, M., & Martini, S. (2021). Computational thinking, between Papert and Wing. *Science & Education*, *30*(4), 883-908. https://doi.org/10.1007/s11191-021-00202-5

Maxwell, J. A. (2013). *Qualitative research design: An interactive approach*. Sage.

Mayring, P. (2015). Qualitative content analysis: Theoretical background and procedures. In A. Bikner-Ahsbahs, C. Knipping, & N. Presmeg (Eds), *Approaches to qualitative research in mathematics education: Examples of methodology and methods* (pp. 365-380). Springer. https://doi.org/10.1007/978-94-017-9181-6_13

McKenney, S., & Reeves, T. C. (2018). *Conducting educational design research* (2nd ed.). Routledge. https://doi.org/10.4324/9781315105642

Mueller, J., Beckett, D., Hennessey, E., & Shodiev, H. (2017). Assessing computational thinking across the curriculum. In P. Rich, & C. Hodges (Eds), *Emerging research, practice, and policy on computational thinking* (pp. 251-267). Springer. https://doi.org/10.1007/978-3-319-52691-1_16

NDET. (2019). *Fagfornyelse* (Curriculum 2020). https://sokeresultat.udir.no/finn-lareplan.html?fltypefiltermulti=Kunnskapsl%c3%b8ftet%202020

Ng, O.-L., & 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. https://doi.org/10.1007/s11858-020-01200-7

Nordby, S. K., Bjerke, A. H., & Mifsud, L. (2022a). Computational thinking in the primary mathematics classroom: A systematic review. *Digital Experiences in Mathematics Education*, *8*(1), 27-49. https://doi.org/10.1007/s40751-022-00102-5

Nordby, S. K., Bjerke, A. H., & Mifsud, L. (2022b). Primary mathematics teachers' understanding

of computational thinking. *Künstliche Intelligenz*, *36*(1), 35-46. https://doi.org/10.1007/s13218-021-00750-6

Nortvedt, G. A., & Buchholtz, N. (2018). Assessment in mathematics education: Responding to issues regarding methodology, policy, and equity. *ZDM Mathematics Education, 50*(4), 555-570. https://doi.org/10.1007/s11858-018-0963-z

Norwegian Centre for Research Data. (2021). *Barnehage- og skoleforskning, NSD Personverntjenester* [Kindergarten and school research, NSD data protection service]. https://www.nsd.no/personverntjenester/oppslagsverk-for-personvern-i-forskning/barnehage-og-skoleforskning

Palm, T., Andersson, C., Boström, E., & Vingsle, C. (2017). A review of the impact of formative assessment on student achievement in mathematics. *Nordic Studies in Mathematics Education*, *22*(3), 25-50.

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

Pajchel, K., Mifsud, L., Frågåt, T., Rehder, M. M., Juuti, K., Bogar, Y., Lavonen, L., Schrøder, V., Aalbergsjø, S. G., & Rognes, A. (2024). Sign of the times: The framing of computational thinking in Danish, Finnish, and Norwegian curricula. *Nordic Journal of Comparative and International Education*, *8*(4). https://doi.org/10.7577/njcie.5744

Rich, K. M., & Strickland, C. (2020). Testing and debugging. In S. Grover (Ed.), *Computer science in K-12: An a-to-z handbook on teaching programming* (pp. 211-218). Edfinity.

Rich, K. M., Yadav, A., & Fessler, C. J. (2022). Computational thinking practices as tools for creating high cognitive demand mathematics instruction. *Journal of Mathematics Teacher Education*, *27*(2), 235-255. https://doi.org/10.1007/s10857-022-09562-3

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. https://doi.org/10.70725/303733vqleui

Saldaña, J. (2016). *The coding manual for qualitative researchers* (3rd ed.). Sage.

Sands, P., Yadav, A., & Good, J. (2018). Computational Thinking in K-12: In-service Teacher Perceptions of Computational Thinking. In M. S. Khine (Ed.), *Computational Thinking in the STEM Disciplines: Foundations and Research Highlights*. Springer International Publishing. https://doi.org/https://doi.org/10.1007/978-3-319-93566-9_8

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

Sneider, C., Stephenson, C., Schafer, B., & Flick, L. (2014). Exploring the science framework and NGSS: Computational thinking in the science classroom. *Science Scope*, *38*(3), 10-15. http://dx.doi.org/10.2505/4/ss14_038_03_10

So, H.-J., Jong, M. S.-Y., & Liu, C.-C. (2020). Computational thinking education in the Asian Pacific region. *The Asia–Pacific Education Researcher*, *29*(1), 1-8. https://doi.org/10.1007/s40299-019-00494-w

Stake, R. (1995). *The art of case study research*. Sage.

Strauss, A., & Corbin, J. (1994). Grounded theory methodology: An overview. In N. K. Denzin & Y. S. Lincoln (Eds), *Handbook of qualitative research* (pp. 273-285). Sage.

Sun, C., Yang, S., & Becker, B. (2024). Debugging in computational thinking: A meta-analysis on the effects of interventions on debugging skills. *Journal of Educational Computing Research*, *62*(4), 1087-1121. https://doi.org/10.1177/07356331241227793

Suri, H. (2011). Purposeful sampling in qualitative research synthesis. *Qualitative Research Journal*, *11*(2), 63-75. https://doi.org/10.3316/QRJ1102063

Tang, X., Yin, Y., Lin, Q., Hadad, R., & Zhai, X. (2020). Assessing computational thinking: A systematic review of empirical studies. *Computers & Education*, *148*, Article 103798. https://doi.org/10.1016/j.compedu.2019.103798

Tikva, C., & Tambouris, E. (2021). Mapping computational thinking through programming in K-12 education: A conceptual model based on a systematic literature review. *Computers and Education*, *162*, Article 104083. https://doi.org/10.1016/j.compedu.2020.104083

Tracy, S. J. (2010). Qualitative quality: Eight "big-tent" criteria for excellent qualitative research. *Qualitative Inquiry*, *16*(10), 837-851. https://doi.org/10.1177/1077800410383121

Ukkonen, A., Pajchel, K., & Mifsud, L. (2024). Teachers' understanding of assessing computational thinking. *Computer Science Education*, 1-26. https://doi.org/10.1080/08993408.2024.2365566

Ukkonen, A., Yadav, A., Pajchel, K., & Xenofontos, C. (2024). Elementary Teachers Assessing Computational Thinking. *Journal of technology and teacher education, 32*(4), 521-546. https://www.learntechlib.org/primary/p/225061/

Wiliam, D., & Thompson, M. (2008). Integrating assessment with learning: What will it take to make it work? In C. A. Dwyer (Ed.), *The future of assessment: Shaping teaching and learning* (pp. 53-82). Routledge.

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

Yadav, A., Hong, H., & Stephenson, C. (2016). Computational thinking for all: Pedagogical approaches to embedding 21st century problem solving in K-12 classrooms. *TechTrends*, 60(6), 565–568. https://doi.org/10.1007/s11528-016-0087-7

Yadav, A., Ocak, C. & Oliver, A. (2022). Computational Thinking and Metacognition. *TechTrends* 66, 405–411. https://doi.org/10.1007/s11528-022-00695-z

Ye, H., Liang, B., Ng, O.-L., & 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*, *10*(1), 3–26. https://doi.org/10.1186/s40594-023-00396-w

Yin, R. K. (2018). *Case study research and applications: Design and methods* (6th ed.). Sage.

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

Attachment A - interview guide (translated from norwegian to English - nuance differences may occur)

Introduction

1. Could you tell me a bit about your background, how long you have been a teacher, and what subjects you teach?

2. What experience do you have with programming, for example through courses or similar?

- In courses or independently of the school/educational context?

Algorithmic thinking as a concept

3. How do you understand the concept of computational thinking?
How do you perceive the difference between programming and computational thinking?

4. Can you describe a lesson you have conducted (or plan to conduct) in which you use computational thinking or programming?

- In which subjects and grade levels?

Planning

5. Can you describe which resources you use in planning?

- Why these resources?

Assessment

6. Can you describe how you have carried out (planned) assessment in this teaching unit? (Refer to the previous answer.)

Assessment of computational thinking

7. In what way do you assess computational thinking/programming?

- How do you develop criteria for assessment?
- What is required for high/low achievement? Why?

8. How do you know whether the students have learned computational thinking and/or programming?

Summative and formative assessment of algorithmic thinking

9. How do you distinguish between formative assessment and summative assessment? Here I am mainly thinking of assessment of programming and computational thinking.

- Which criteria are important in formative and summative assessment?
- Why?

Curriculum

10. If the teacher does not mention the curriculum:

How do you perceive that the curriculum emphasizes assessment in computational thinking and programming?

- In what way do you use the curriculum in the work with assessing programming and computational thinking?
- How do you use the competence aims?

Personal experiences

11. Why do you think computational thinking and programming have been given a place in the curriculum?

- Do you have any personal views on this?

12. Do you see any challenges in assessing computational thinking and/or programming?

- What is needed to solve these?
- Are there any steps you take to address these challenges?

Conclusion

13. Is there anything you would like to add before we end the interview?