# Computational thinking and mathematics education: Debating synergies and tensions

Ivan Kalaš[1], Iveta Kohanová[2],
Piers Saunders[3], Alison Clark-Wilson[3], and Eirini Geraniou[3]

[1]Comenius University in Bratislava, Slovakia
[2]Norwegian University of Science and Technology, Trondheim, Norway
[3]University College London, UK

*This paper records the Plenary Panel discussion held at the third conference on Mathematics Education in the Digital Era, which was held from 7-9th September 2022 in Nitra, Slovakia. The panel discussion, which was chaired by Eirini Geraniou, invited the three panelists (Ivan Kalaš, Iveta Kohanová and Piers Saunders) to give perspectives on Computational Thinking (CT) in relation to mathematics education from three different country perspectives (Slovakia, Norway and England). The discussion addresses important differences between computational thinking and mathematical thinking, the challenges associated with the design and assessment of curricular, and the implications for teachers and their ongoing professional learning. It concludes with a reaction by Alison Clark-Wilson, who highlights the importance of epistemologically grounded design principles for new curricular, the related technologies, tasks and assessments.*

*Keywords: Computational thinking, computer science, computing education, algorithmics, informatics.*

## Introduction

Over the last 10 years or so, Computational Thinking (CT) has become increasingly evident in both mathematics education research and mathematics teaching practice. This is due to new curricula that emphasise CT as an important 21st century skill for learners. Also, the literature is characterising CT as an essential competency for a digital society (Inprasitha, 2021) or the "new digital age competency" (e.g., Grover & Pea, 2013). However, education in relation to CT is implemented differently across Europe. In some countries, it is a compulsory subject, for example, informatics, computing, or computer science. In other countries it has become part of the mathematics curriculum, or integrated within a combined set of curriculum subjects within a broader STEM or STEAM curriculum. However, although there is growing literature on CT in mathematics education or other disciplines, there exist many different perceptions and expectations of the potential of CT for mathematics education, within and beyond mathematics education researchers, teacher educators and teachers.

This paper reports the panel discussion in which Ivan (Slovakia), Iveta (Norway) and Piers (UK) offer their views on CT within the context of mathematics education. They drawi on examples from their respective countries. The panel, which was chaired by Eirini, was framed by three questions, each of which was directed to one panelist first, followed by a short response from the remaining two panelists. This was then followed by Alison's reaction.

## Panel Question 1: Without attempting to define either computational thinking (CT) and/or mathematical thinking (MT), what differences, if any, do you see between them? If yes, can you characterise one such difference?

### Response 1 by Ivan

Let me start by pointing out that mathematics education has always been, and continues to be a great inspiration for me in my work to develop educational content for informatics. This is not only through its long tradition of looking for appropriate content and pedagogy, but also by its complexity and sophisticated progression, from year to year, and from school phase to school phase. The field of Informatics education has a lot to learn from mathematics education. In informatics education, we are still busy clarifying content and only just beginning to explore the cognitive demands and appropriateness of particular concepts; learn to distinguish digital literacy from computer literacy; and so on. What has become established, is an understanding of the relationships between computational thinking and programming.

If we carefully consider mathematical thinking, computational thinking and programming, we very soon find a common concept in the background, namely the algorithm. This is present in informatics itself, and equally when we examine the role that CT plays in the development of MT. Nevertheless, I believe that there are significant differences in the two educational perspectives on both algorithms and when solving problems.

As I do not dare, nor attempt, to define either CT or MT, I will try to present my perceptions of the big goals for informatics and programming as we design and implement educational content in the school context. I do this through an example from the lower primary curriculum for pupils in Years 3 and 4 (8-10 years old).

In Slovakia, informatics has been a separate compulsory subject for lower secondary students since 2004, and for primary age students since 2012, which is 10 years now. The curriculum begins in Year 3 and continues in every subsequent year (with one exception) until the end of K12 education. However, although the formal curriculum begins in Year 3, many schools introduce informatics even earlier. In our design research group, we have developed non-mandatory informatics educational content for kindergarten (5-year-olds). An example of one of our pre-programming environments Emil (named after the featured character that provides the familiar narrative for the children) is shown in Figure 1.

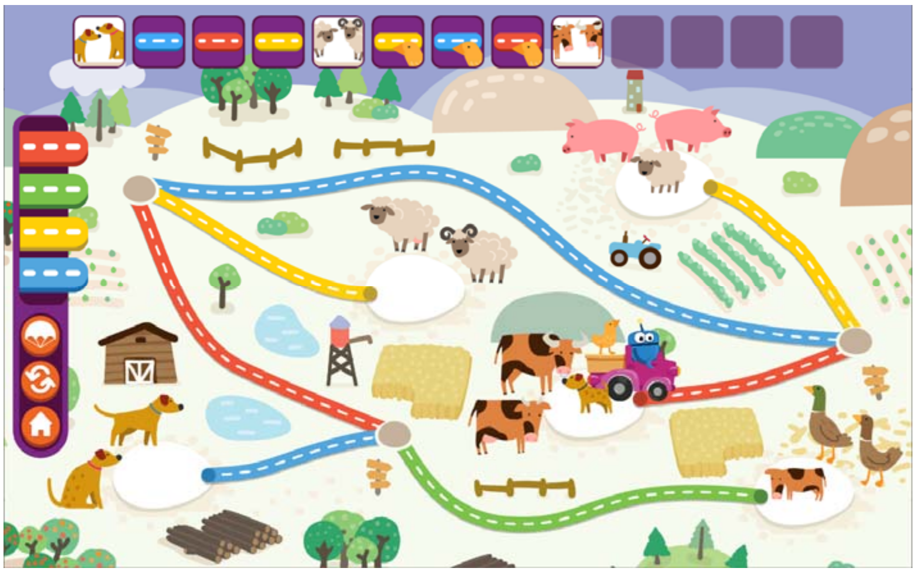


**Fig. 1: Introductory activities for kindergarten children, designed for a group of 4 to 6 pupils, to work collaboratively in front of the IWB.**

In this example, a group of 4 to 6 children would work collaboratively within this environment in front of the IWB. Presented with a sequence of situations, they work to solve a problem on a map with coloured paths, guiding the character "Emil" in his small truck by entering the colour of the path he should follow. In this example, the children work to instruct the path Emil should take to collect the lost animals (a little goose, puppy and calf) and return them to their families.

At the same time, a symbolic record of the steps is created on the top line of the IWB screen. In our task designs, even at this age, we always follow a gradual cognitive transition from directly controlling an actor and creating a record of our steps, to planning the steps while using the same symbols to express these steps. We call this process step planning, and its representation in a particular language programming. Intentionally and specifically, I emphasise the notions of (1) creating a record of steps, (2) planning steps to solve a problem, and (3) a language in which we represent both the record and the plan (the programme).

Indeed, programming plays a key role in both our educational content development and our related research on programming concepts, which concern the cognitive demands for pupils at different years and stages. We view programming very broadly, by considering it in a sense as the language of informatics, as a means and a tool of computational thinking, in all its components. To be more specific, these components are usually recognised to be abstraction, algorithmic thinking, decomposition, generalisation, and evaluation. Consequently programming is broadly understood not as a goal but as a tool – in this way it has the potential to play an important role in supporting pupils to develop each of these components, whilst providing them with opportunities to explore, model, express, and collaborate etc.

Blackwell says that pupils start programming when they stop directly manipulating observable things but specify behaviour to occur at some future time (Blackwell, 2002). And why is this so challenging?

Because, instead of reaping the benefits of direct manipulation, we introduce notational elements to represent behaviour, abstraction, and change. In doing so, we run into various constraints. These relate to the language used, the representation of the program, and in the behaviour of the character we are controlling, in the context of the actual problem, let me illustrate this with a second example, that introduces Emil and Ema, the virtual and floor programmable robots we use in Slovak and Czech primary schools.

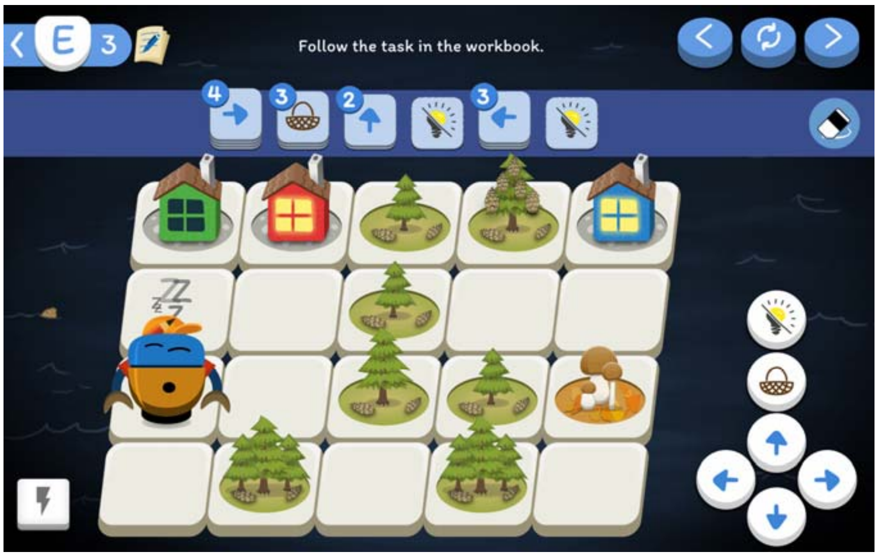


**Fig. 2: An example that develops the notion of the programme as a set of instructions to be enacted in the future.**

In Figure 2, we see: Emil's scene (or world); a group of commands and tools (right) for controlling his movements and applying his tools to the scene cells; and a blue panel (top) for building the program. When the pupils have finished planning, they wake Emil up to run their program.

As the content is so extensive, I am not able to give a complete picture. Instead, I highlight some of our specific design principles and some important milestones when moving from direct control to programming. First, in our pedagogy, primary pupils never work ‘one-to-one’ with computers, but always in pairs. When they work with robotics (i.e. with Ema), they are in even larger groups. When working with Emil, two pupils have one shared tablet or laptop and two workbooks in front of them. Together they discuss and solve sequences of problems that are ordered by increasing cognitive demands. Second, what is unique about our approach – at least when compared to many schools in our countries – is that there is no space for any traditional teaching in our classes, by which I mean ‘lecturing’. The teacher never reveals a new concept or procedure. The sequence of tasks is designed so that the pupils explore and discover everything for themselves, through collaboration and constant discussions, which are guided by the teacher with proper questions. Pairs of pupils share, explain, compare, and justify their strategies and solutions to each other. In that way they construct deep and durable understanding. Finally, the third design principle I want to mention is the fact that our programming environments of Emil for Years 3 and 4 do not give any feedback. It is the pupils themselves who have to consider and assess their progress – first in their pairs, but also many times a lesson as a whole group.

Concerning the progression in the content, let me jump straight to an activity involving Emil in the middle of the progression for Year 3. By this stage, pupils are already familiar with the fact that when a problem begins with Emil asleep, this signals that they have to program the solution first, and only then wake Emil as they are ready to run (or execute) their program.

In earlier activities, pupils have already encountered a significant restriction in the form of the length of the blue panel, i.e. the number of steps of their solution is restricted by the number of positions on the blue panel. In Figure 3, for example, we see a pupils' solution to one of the problems. It is this restriction that makes some tasks unsolvable (while others have multiple solutions), others are open-ended or even 'unclear' in the sense that pupils are invited to formulate their own additional rules. These constraints are all the subject of extensive whole class discussion.

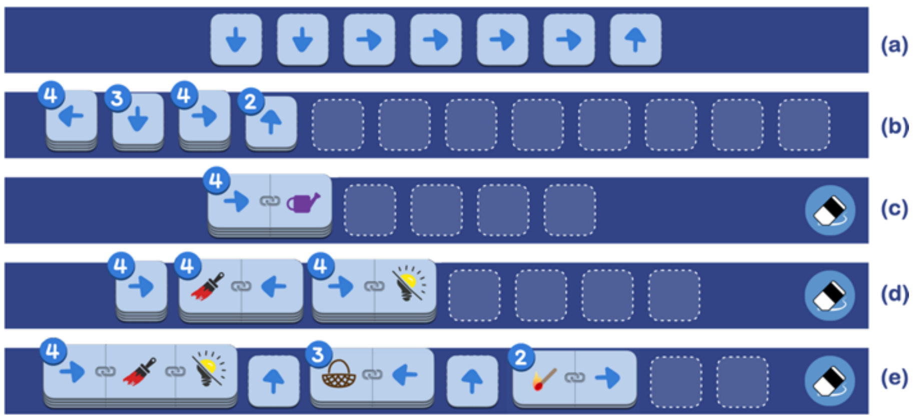


**Fig. 3: Different panel designs to show the program provide opportunities to think about its structure and properties.**

The blue panel can have a maximum of 12 positions, but for most tasks this number is often lower. For example, in the case of the task relating to Figure 3 (a), up to seven steps can be planned. However, in subsequent problems in the progression, pupils discover new panel behaviour: If they enter the same command multiple times in a row, the icons in the panel automatically stack up, see the solution in panel (b). Even though this program consists of 13 commands, it takes only four positions on the panel. We increasingly transfer pupils' attention to the panel. They consider and discuss various properties of their program, such as its structure. Thus, they discover that two consecutive commands can be connected into a double command. If they create identical double commands in the program in a row, these also automatically stack up, see panels with programs (c) and (d). Three cards can be merged into one as well. So, the fifth program (e) consists of up to 24 commands, but we have created it in panel (e) with only 11 positions, taking just nine of them. Thus, we promote pupils' understanding of the elements of repetition in the program.

In Emil for year 3 we control the character in a so-called absolute frame, that is, with the arrows up, right, down, and left. In Emil for Year 4, we move further in the sense that we control the character in a so-called relative frame, that is, with the commands step forward and turn right 90° or left 90°. Within the activity sequences we gradually add more and more language commands, see Figure 4, so that pupils can always focus on discovering new possibilities and their properties. We start with a simple pair of commands: step forward and stamp the green star (and later add a step forward with drawing a line). Gradually, pupils will discover how they can choose the colour of the pencil and stamp, and the thickness of the crayon or the size of the stamp. They will also begin filling the enclosed area with a selected colour that they have previously outlined. They will also discover how to use the pin command to draw diagonal lines in a regular square grid, where Emil moves only on horizontal and vertical grid lines.

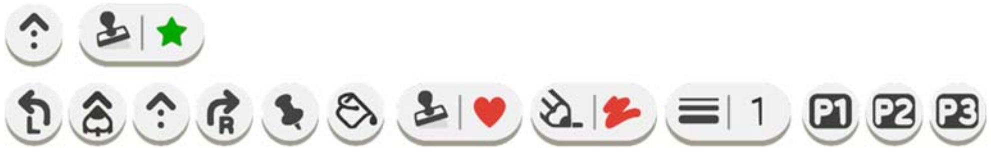
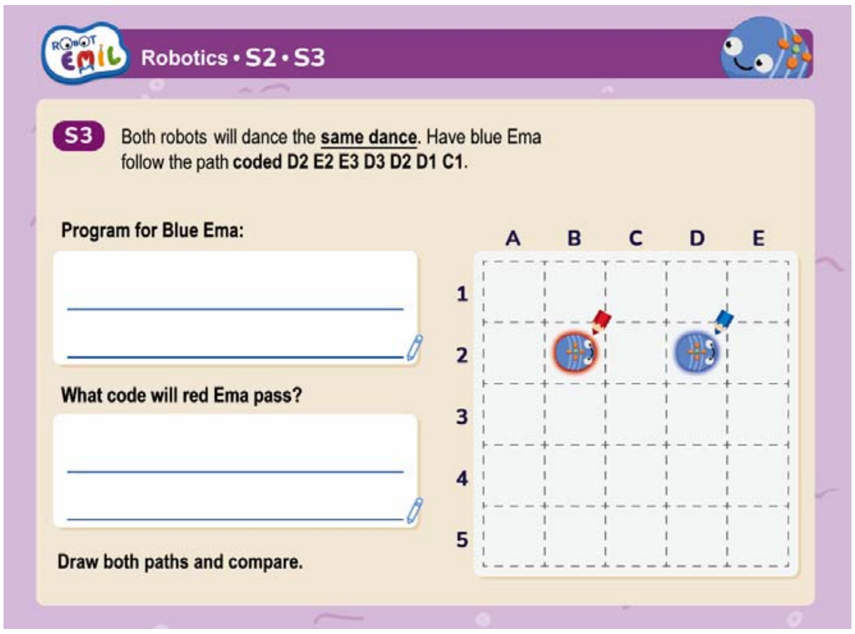


**Fig. 4: In Emil 4, we pupils start with only two simple commands**

In addition to a vocabulary of nine basic commands, Emil 4 also provides three compound commands P1, P2 and P3. First, these are defined in the tasks by the authors. Pupils explore them and use them together with other basic commands as shortcuts for groups of commands that logically belong together (in the sense of new blocks in Scratch or user-defined functions in Python). Later on, pupils start to modify and correct provided definitions. Only then do they begin to create and use their own compound commands. We consider this as an important development in the level of pupils' abstraction in their computational thinking.



Both robots will dance the same dance. Have blue Ema follow the path coded D2 E2 E3 D3 D2 D1 C1.

Program for Blue Ema:

What code will red Ema pass?

Draw both paths and compare.

**Fig. 5: In Robotics with Ema we focus on multiple representations of different things. In this case we are using a special and yet intuitively simple representation of the path**

Before I try to summarise why and how these short vignettes illustrate what I consider to be special about computational thinking when viewed from an informatics perspective, I want to mention another part of our educational content in primary informatics. These activities are implemented by schools within five lessons in each primary year 1 to 4. This is a specific comprehensive progression of activities in which pupils work on the floor in groups of three to four with special mats and Blue-Bots, who we have named as a female robot, Ema. One of the main goals of this progression of activities is to introduce multiple representations. These include how we identify the different squares on the mat, how we represent Ema’s position and direction, how we represent the program; how we control Ema, how we represent the corresponding path on the mat, what shape that path is, etc. In Figure 5, we see an activity from Year 4 where we work with a mat based on a 5 x 5 array. We label its squares in a familiar way, e.g., B2, C2, D4... We name Ema's position and direction, e.g., by saying, Ema is standing on B2 facing away from A2. When Ema executes her program from the start and with the initial direction, she walks a certain path on the table mat: the program corresponds to the path on the mat and the path’s shape. And can also be expressed by the sequence of the labels of the squares she walked on. In the activity in Figure 5, we see that blue Ema has to take a path with the ‘code’ D2 E2 E3... What will be the code of the red Ema’s path if both robots execute the same program?

So in conclusion, where do I see differences between MT and CT? This is a hard question, which I have tried to answer in an indirect way by showing what we do within the context of primary school informatics. We solve contextual problems by controlling a character or several characters. We emphasise how they can be controlled and how we can represent that process. We explore how the characters behave, how they react to different events and situations, what their options are and what their constraints are. These interest us not only in the characters’ behaviour, but also at the level of data, in several different senses. In accordance with Papert, we try to get pupils to think about the program itself, to consider it as an expression of their idea, to explore different properties of the program, such as its length etc. Pupils compare different representations of a procedure for solving a problem, exploring the language used to represent it. They compare different solutions and explore whether they would be able to express the solution if certain constraints were added, if they had other means and structures to engage etc. We want pupils to encounter different powerful ideas of computing in this way. In computing. And in mathematics as well.

**Response 2 by Iveta**

I will provide some examples from a survey we conducted in the spring of 2022 of around 350 Norwegian mathematics teachers (Turgut et al., 2022). The survey was framed by the Pedagogical Technology Knowledge framework (Thomas & Palmer, 2014), with a focus on the implementation and use of tools for computational thinking and programming (CTaP tools) in mathematics teaching.

One of the open questions was related to the teachers’ perceptions of the effectiveness of these tools for the purpose of supporting their students to have an improved understanding of mathematics. The analysis of the responses revealed teachers’ views that imply some differences between computational and mathematical thinking.

Teacher 1: CTaP tools provide immediate feedback. You get the results quickly, no need to wait for the teacher and you see whether your solution is correct or not correct. So you are naturally motivated to look for an error which is not happening in mathematics. Thus, students learn from their own mistakes.

Teacher 2: I think the students get a feeling when using these tools that mathematics should be used to arrive at a solution or to create something. Therefore, they must understand that mathematics is a tool to get the desired result. [and] In a traditional problem in mathematics textbook, I don't think they get the same understanding … that we use mathematics in order to create something.

Teacher 3: It’s a borderland between theoretical and practical mathematics because it embraces both worlds in a good way and this can lead to flourishing in other students than those who did well in classical mathematics.

Teacher 1’s response indicates that evaluation and debugging are one of the differences between CT and MT. The teacher’s example is closely related to programming. Also, the teacher refers to trial and error which is a common heuristic in programming but for many students seen as somewhat invalid in traditional mathematics. In addition, students might be motivated by having less fear of making errors.

The CTaP tools, according to Teacher 2, help students see the usefulness of mathematics. This can be connected to problem solving or entrepreneurial activities. In mathematical thinking it is harder for students to see how their answers and solved problems can be useful for real life purposes.

Teacher 3 sees opportunities for underachieving or underperforming students, as well as for students with a lack of motivation in mathematics. The practical nature of programming found in, for example, debugging and the creation of algorithms, could appeal to many of the students who previously lost interest in mathematics.

In winter 2021, we asked Norwegian in-service mathematics teachers participating in a professional development course, to draw mind maps and compare components of mathematical and computational thinking. The analysis of their mind maps revealed that generalisation, abstraction, analysis and problem solving were the components that were typical for both MT and CT. Components which appeared only in the MT part were reasoning and proof and communication. On the other hand, components present only in the CT part were expressed by verbs related (mainly) to programming, like debug, structure, document, sort, declare, and decompose. But also, to google, to try and to cry. This indicates that Norwegian mathematics teachers perceive/interpret CT as programming and/or coding, which is one of the findings of Nordby and her colleagues (2022) as well. The verbs “trying” and “crying” might express teachers’ uncertainty and confusion, which signal a need for professional development courses related to CT. We have noticed a similar issue in the survey answers (Turgut et al., 2022), which resonate with findings from the studies by Kveseth (2022) and Grimsgaard (2022).

## Response 3 by Piers

With respect to differences and similarities, my thinking is very much inspired by Cuoco’s (1996) “Mathematical habits of mind” which I wrote about and discussed in some depth in my recently completed doctoral thesis (Saunders, 2022). I really feel those habits of mind have strong similarities with how computational thinking has been defined within the literature. For example, if you're not familiar with these, Cuoco describes a series of statements such as “students should be pattern sniffers” or “students should be tinkerers”. I think we can see very, very strong links with aspects that Ivan talked about within his presentation for the types of activities that we want students to be doing as they engage with computational thinking, whilst also being the types of activities that we want students to do as they develop mathematical thinking.

## Panel Question 2: Do the new curricula look different within the primary, secondary and tertiary school phases? For example, are the digital tools used in each phase the same or different? Are there any particular implications for assessing learners' outcomes?

### Response 1 by Iveta

In Norway we have a new national curriculum since August 2020 in which CT and programming are introduced in the following subjects: mathematics, science, arts and crafts, and in music. Computational thinking is in Norwegian translated as "algoritmisk tenkning" (algorithmic thinking) and thus it inevitably leads to misunderstandings and misconceptions, as the term *algorithm* has associations with standard algorithms (Gjøvik & Torkildsen, 2019; Nordby et al., 2022).

CT is mentioned in the mathematics curriculum for Grades 1-13[1] only once, under core element *Exploration and problem solving*.

> "CT is important in the process of developing strategies and approaches to solve problems and means breaking a problem down into sub problems that can be solved systematically. This also includes evaluating whether the subproblems can be solved best with or without digital tools."
>
> (Directorate of Education, 2019a, p.2)

On the other hand, the curriculum uses the term *programming* often. The general idea is that students learn different terms and concepts related to programming in mathematics, which they they apply within mathematics, science, arts and crafts and in music (Sevik & Guttormsgaard, 2019). For example, one of the competency goals in the music curriculum for Grade 10 says: "Create and program musical sequences by experimenting with sounds from different sources" (Directorate of Education, 2019c, p. 8).

Coming back to mathematics, there is one competency goal (of the 10-15) related to programming in the mathematics curriculum for each grade (starting from year 2). Some of them are very general, neither specifying the programming language and (digital) tools to use, nor the mathematical topic. For example, a competency goal within Grade 5 states, *"create and programme algorithms with the use of variables, conditions and loops"* (Directorate of Education, 2019a, p. 9). In Grades 3, 6, 7 and 9 only the mathematical topics are specified:

· create and follow rules and step-by-step instructions in play and games related to the coordinate system

· use variables, loops, conditions and functions in programming to explore geometric figures and patterns,

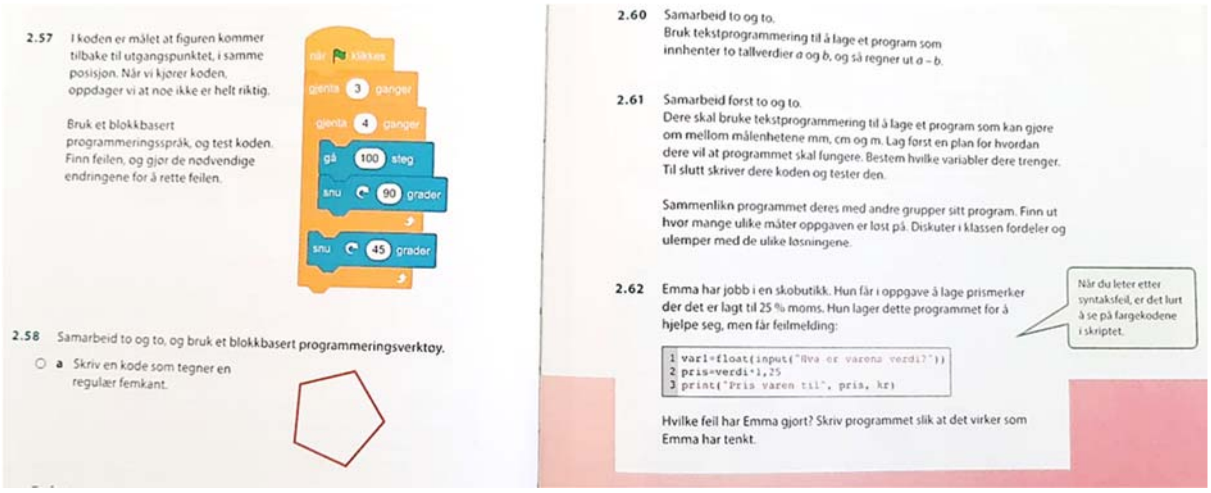
· use programming to explore data in tables and datasets,

[1] In Norway, children start Grade 1 the year they turn six years old.

- simulate outcomes in random events and calculate the probability that something will occur by using programming.

(Directorate of Education, 2019a)

Similar to the curriculum, some textbooks do not specify the programming language. However, from the text (Figure 6), although not explicitly stated, it can be inferred that the languages used are Scratch or Python. From the aforementioned teachers' survey responses (Turgut et al., 2022) we also know that CTaP tools like MakeCode with the Micro:bit are often used in Norwegian schools, as well as Scratch, Python, GeoGebra or spreadsheets in Grades 6-10. In Grades 2-5 we found that Bee-bot, Scratch, Robot Emil or various online environments are used regularly.



2.57 I koden er målet at figuren kommer tilbake til utgangspunktet, i samme posisjon. Når vi kjører koden, oppdager vi at noe ikke er helt riktig.

Bruk et blokkbasert programmeringsspråk, og test koden. Finn feilen, og gjør de nødvendige endringene for å rette feilen.

```
når 🏁 klikkes
gjenta 3 ganger
  gjenta 4 ganger
    gå 100 steg
    snu ↻ 90 grader
  snu ↻ 45 grader
```

2.58 Samarbeid to og to, og bruk et blokkbasert programmeringsverktøy.

a Skriv en kode som tegner en regulær femkant.

2.60 Samarbeid to og to.
Bruk tekstprogrammering til å lage et program som innhenter to tallverdier *a* og *b*, og så regner ut *a* – *b*.

2.61 Samarbeid først to og to.
Dere skal bruke tekstprogrammering til å lage et program som kan gjøre om mellom målenhetene mm, cm og m. Lag først en plan for hvordan dere vil at programmet skal fungere. Bestem hvilke variabler dere trenger. Til slutt skriver dere koden og tester den.

Sammenlikn programmet deres med andre grupper sitt program. Finn ut hvor mange ulike måter oppgaven er løst på. Diskuter i klassen fordeler og ulemper med de ulike løsningene.

2.62 Emma har jobb i en skobutikk. Hun får i oppgave å lage prismerker der det er lagt til 25 % moms. Hun lager dette programmet for å hjelpe seg, men får feilmelding:

```
var1=float(input("Hva er varens verdi?"))
pris=verdi*1,25
print("Pris varen til", pris, kr)
```

Hvilke feil har Emma gjort? Skriv programmet slik at det virker som Emma har tenkt.

Når du leter etter syntaksfeil, er det lurt å se på fargekodene i skriptet.

**Figure 6. Programming in a year 8 mathematics textbook (Tofteberg et al., 2020, p. 136-137)**

In upper secondary school, which in Norway it means Grades 11-13, mathematics is compulsory only in year 11 and students can choose between practical (P) and theoretical (T) mathematics. Only students who choose theoretical mathematics will further develop their competence related to CT and programming, as the curriculum states:

"formulate and solve problems through the use of computational thinking, different problem solving strategies, different digital tools and programming."

(Directorate of Education, 2019b, p. 5)

Any application related to a mathematical topic is specified only in Grade 13, *"develop algorithms to calculate integrals numerically, and use programming to execute the algorithms"* (Directorate of Education, 2020, p. 6).

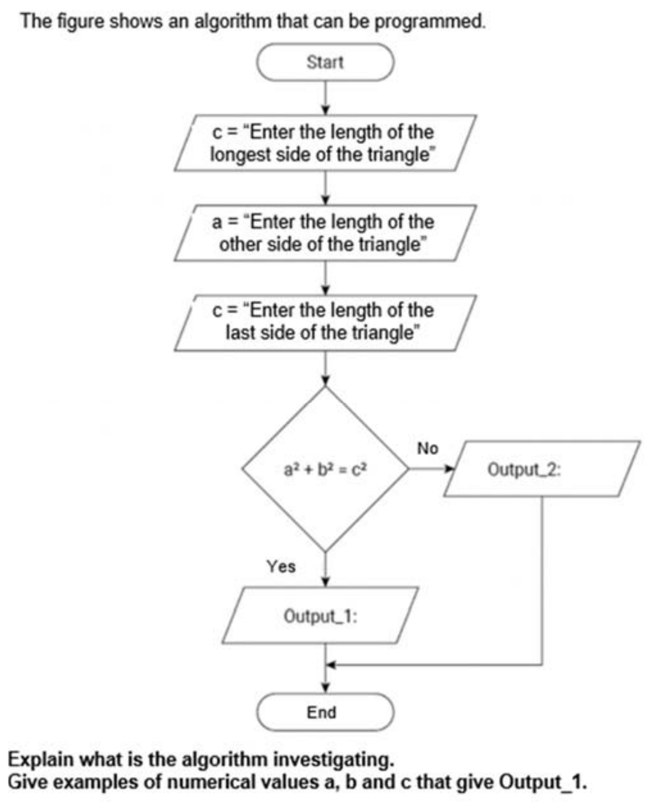
As we can see from the above, in mathematics students learn programming as well as the application and use of programming. CT seems to be under-communicated in the curriculum and there is also some confusion regarding what it is, and its place in mathematics.

Moving to the implications for assessing learners' outcomes with respect to CT, it is important to mention that pupils in Norway are not given grades or marks during the first seven school years. Instead, at the end of each term, students are assessed summatively (against competency standards), together with a guidance on how s/he can increase his/her competence. From Grade 8 onwards, students are given grades and must also have a written half-year assessment with a grade. Assessment practices have traditionally been weak, with teachers focusing on effort rather than the quality of students' competence and curriculum mastery (Hopfenbeck et al., 2012). Since 2007, national mapping tests in reading and numeracy have been implemented to help primary school teachers identify the weakest 20% of students (Nortvedt et al., 2016). In addition, during Grades 5, 8 and 9 the schools are given knowledge about their pupils' basic skills in computing/calculating from national testing. To date, none of the tasks in these national tests relate to CT or programming, which seems fair since these ideas have only been part of the national curriculum since autumn 2020, and teachers need some time to adjust their teaching accordingly. However, there is little recent research on Norwegian primary school mathematics teachers' classroom assessment practices (Nortvedt et al., 2016). I hypothesise the same would be true for secondary school teachers. Again, the new curriculum has only been effective for about two years (at the time of writing) so the research gap becomes even bigger in relation to the assessment practices within CT and/or programming. It will also be interesting to study how mathematics teachers' understanding of CT affects their assessment practices in mathematics.

My final point concerns the national exams (high-stakes tests) in mathematics. In the *example set*[2] for the Grade 10 exam for 2022, there was one task in which students were asked to explain an algorithm (displayed as a flowchart shown in Figure 7) and give examples for numerical values for which a certain equality holds. The context of the task was mathematical. The purpose of the algorithm was to determine whether the input of a triplet of triangle side lengths (a, b, c) satisfy the Pythagorean theorem (inferring that the sides represent a right-angled triangle).

A second open task was also included in the example set (Figure 8).

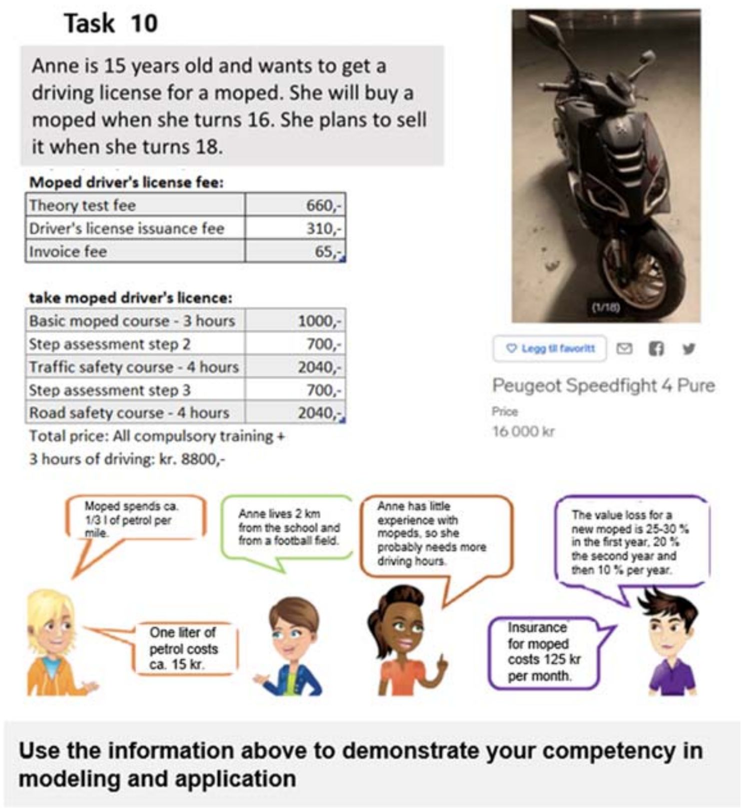
[2] An example set is a complete exam distributed to teachers but only for training and information purposes.


The figure shows an algorithm that can be programmed.

Start
c = "Enter the length of the longest side of the triangle"
a = "Enter the length of the other side of the triangle"
c = "Enter the length of the last side of the triangle"
$a^2 + b^2 = c^2$
No
Output_2:
Yes
Output_1:
End

**Explain what is the algorithm investigating.**
**Give examples of numerical values a, b and c that give Output_1.**

**Figure 7: Flowchart from Grade 10 example set (Directorate of Education, 2022a)**


**Task 10**

Anne is 15 years old and wants to get a driving license for a moped. She will buy a moped when she turns 16. She plans to sell it when she turns 18.

**Moped driver's license fee:**

| | |
|---|---|
| Theory test fee | 660,- |
| Driver's license issuance fee | 310,- |
| Invoice fee | 65,- |

**take moped driver's licence:**

| | |
|---|---|
| Basic moped course - 3 hours | 1000,- |
| Step assessment step 2 | 700,- |
| Traffic safety course - 4 hours | 2040,- |
| Step assessment step 3 | 700,- |
| Road safety course - 4 hours | 2040,- |

Total price: All compulsory training + 3 hours of driving: kr. 8800,-

Peugeot Speedfight 4 Pure
Price
16 000 kr

Moped spends ca. 1/3 l of petrol per mile.
Anne lives 2 km from the school and from a football field.
Anne has little experience with mopeds, so she probably needs more driving hours.
The value loss for a new moped is 25-30 % in the first year, 20 % the second year and then 10 % per year.
One liter of petrol costs ca. 15 kr.
Insurance for moped costs 125 kr per month.

**Use the information above to demonstrate your competency in modeling and application**

**Figure 8: Open task from Grade 10 example set (Directorate of Education, 2022a)**

The accompanying task guidance stated:

In this assignment, you are expected to:

- ask relevant mathematical questions that demonstrate your competence
- show calculations and answer your questions
- make critical assessments based on the questions and your calculations

In the solutions to their own questions, students are expected to demonstrate their competence in *modeling and applications*, another of the core elements of the new curriculum in Norway. In the suggested solution provided by the Norwegian Directorate for Education (2022b), the expected solution is to use a spreadsheet. It can be argued that the creation of such a spreadsheet also can be considered as being within the scope of CT.

The flowchart task was presented without the specification of any programming language. This might happen in later examination designs, but for now it seems that the choice of language, or even whether to choose unplugged, block-based or textual programming, is the responsibility of the teachers and the schools. The flowchart task might seem fairly easy, but this should be seen as a first step into programming in mathematics in Norwegian schools. The open task (Figure 8) demonstrates another new, very open, type of task. This task is representative of some of the new content and new core elements in the curriculum. These new types of tasks will push teachers into making several adjustments to their teaching, and potentially face several challenges. For example, how should they teach the new topics? They also have to consider new types of assessment practices connected to the new task types so that students taking the exams will get fair assessments in these new experiences.

**Response 2 by Piers**

One key point which continues to perplex me is that when we are thinking about the vehicle of programming for learning mathematics, we also need to address the assessment practices of that learning. For example, Iveta talked about programming through the different curricula without focusing on the specific language such as Scratch or Python itself. She also highlighted in her examples that when we assess programming in the high stakes (end of phase) examinations, the method is through pencil and paper tasks. But surely, the role of programmers is to actually program with a computer! If we consider the role of digital tools in mathematics in the UK, the current assessment structures continue to be pencil and paper based with *some* access to a calculator, although there is a non-calculator examination paper! And so I pose the important question, “How do we bring about system change in the high stakes assessment practices so that digital tools (beyond calculators) are at first recognised and then permitted? ”

**Response 3 by Ivan**

Again, I will use school mathematics as a parallel. A curriculum for developing computational thinking and programming needs to be complex and comprehensive for the whole of education, across all stages of school, with clear goals, vision and direction. However, it must systematically progress from year to year, respecting developmental appropriateness, interests of the pupils and the level of their abstract thinking. Motivating them appropriately, building on what we have learned earlier. Otherwise, we would get stuck in a loop, accepting vague learning goals and starting all over again in each stage.

In this process, primary school plays a special role. Primary teachers teach most of the subjects to their pupils, so we must not expect them to be specialists in informatics and programming. But this is not a limitation. On the contrary, it is an amazing advantage! Primary teachers have big experience in how to bridge learning between subjects and various learning areas. We need to build the curriculum in informatics and programming to respect and utilise this primary learning ecology.

## Panel Question 3: What are the implications of such curricula developments for teacher preparation and support? Are there any local, regional or national initiatives to address teachers' professional learning needs?

### Response 1 by Piers

Both Ivan and I worked on the ScratchMaths project which positioned building mathematical knowledge through learning to computer program. In the UK a new compulsory computing curriculum was introduced in 2014, for all phases of education, from children as young as 5 years old up to 16 years of age. Ivan has stated that we need to respect primary teachers, and so when the computing curriculum was introduced, we were very aware that primary school teachers would not be prepared for teaching such a curriculum. We therefore had an opportunity to design a new curriculum for both the teachers *and* the children, and set an overarching goal to develop problem solving and reasoning in mathematics through programming in Scratch. The success of our project was evidenced by an independent evaluation, which measured the children's mathematics scores on a national test at the end of the primary phase aged 11 years old. The project was a large national project aimed at children who were aged 10 and 11 years old, i.e. the final two years of the primary phase in England. Our design needed to reflect the UK context, in particular the background of children, and the technology that they had available to them at school. Importantly, this was not an initiative for children who had previously experienced programming, and likewise the teachers. The children and the teachers were learning and starting from *Scratch* as they engaged with our carefully sequenced two year ScratchMaths curriculum (Figure 9).

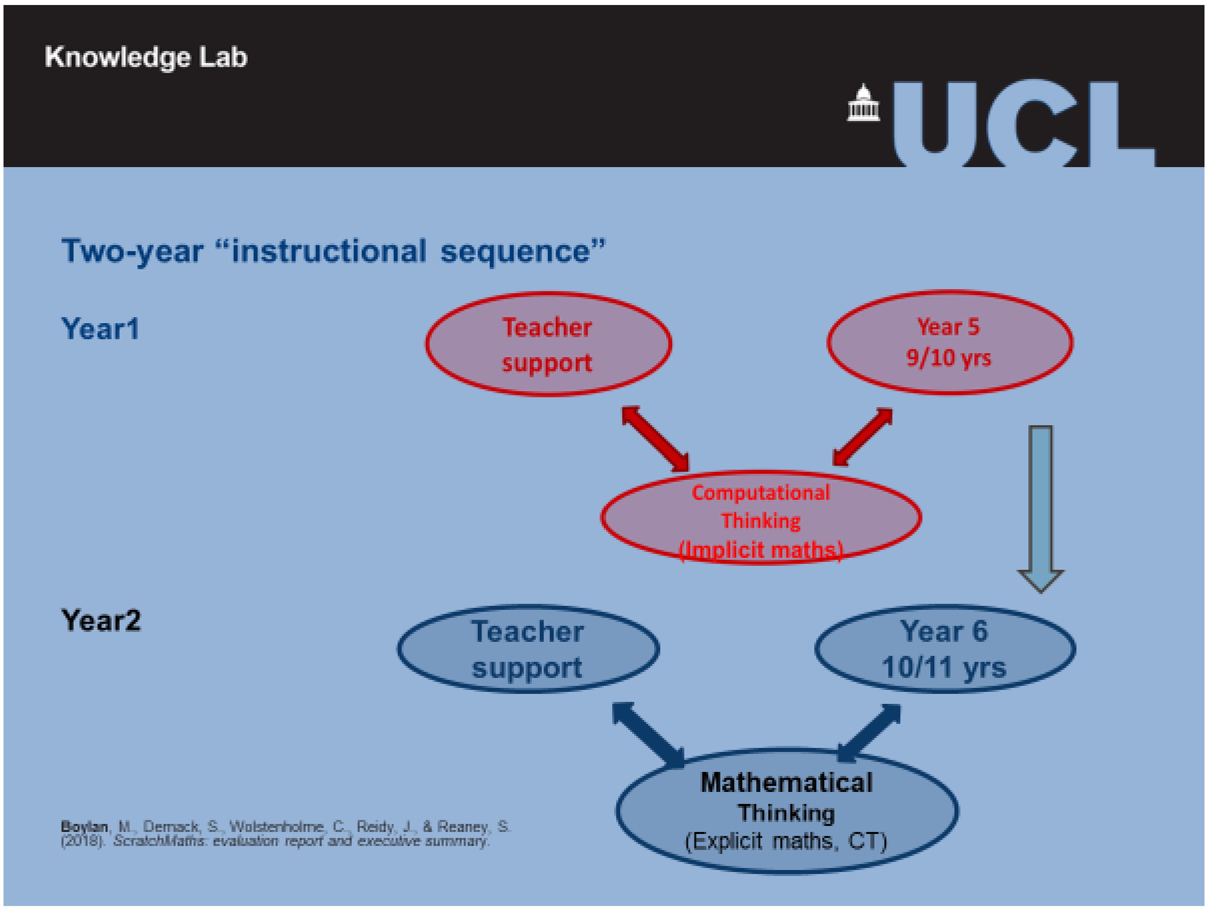

Knowledge Lab
UCL
Two-year "instructional sequence"
Year1
Teacher support
Year 5 9/10 yrs
Computational Thinking (Implicit maths)
Year2
Teacher support
Year 6 10/11 yrs
Mathematical Thinking (Explicit maths, CT)
Boylan, M., Demack, S., Wolstenholme, C., Reidy, J., & Reaney, S. (2018). ScratchMaths: evaluation report and executive summary.

**Figure 9: The ScratchMaths instructional sequence across the two years of the curriculum (Boylan et al, 2018).**

In year one we provided teacher professional development, through in-person training accompanied by detailed curriculum materials (freely available at http://www.ucl.ac.uk/scratchmaths): pupil tasks, teacher guides, posters and Scratch starter projects. The focus of the ScratchMaths curriculum for children aged 9 was on developing computational thinking or aspects of computational thinking where the mathematical concepts were *implicit*. For example, in the first module, ideas of movement, rotation and pattern are explored by controlling the *sprite*[3] with programming commands and structures. Students engage with the coordinate system within Scratch as they notice the numbers changing as they move the sprite forwards and backwards. Thus, in the first year of the curriculum the children (and teachers) build a foundation of programming skills. In the second year of the curriculum, we provided additional professional development for the teachers. The children were now a year older, the teachers had experienced teaching Scratch and ScratchMaths for a year and they were now able to develop mathematical thinking and mathematical reasoning where the maths is very, very explicit.

The ScratchMaths final report was published in 2018 by an independent evaluator who found that there was a positive and significant effect on computational thinking skills in Year 5 (Boylan et al,

[3] the programmable object in Scratch

2018). For us this was really important since the teachers had received appropriate professional development,  taught the curriculum materials as planned, and had the same experiences as the children. However, in the second year of the curriculum when the children were a year older, not all teachers moved forward with the children. Consequently, we were not surprised that the mathematics outcomes at the end of the second year (as measured by the national test) were not significantly affected for those that had studied ScratchMaths. Teachers not following their class as the children progress to the final year in the UK primary system is quite common as schools often timetable mathematics classes with a specialist mathematics teacher to teach and support children towards the national examination. We therefore had a context where the teachers were teaching the second year of a curriculum, having not had any experience of the foundational aspects of the first year.  Ivan often uses the analogy of trying to learn (and teach) a foreign language when you have only been given the year 2 textbook. So with respect to implementing a new curriculum successfully, it is essential to understand the teachers' context and how they progress from year to year.

However, during the project we worked directly with teachers in London who acted as design partners. This enabled us to learn that providing only curriculum materials was not enough for a successful implementation. We also needed to support teachers with how to teach programming which needed a different pedagogy. Consequently, we developed a pedagogical framework which you may have heard us talk about before, called the 5E Pedagogical framework (Noss et al, 2020). The framework was embedded throughout the curriculum materials to support the teaching of computational thinking and programming ideas, and to explore teaching mathematics through programming. Explain, Explore, Envisage, and Exchange are fairly well articulated pedagogical approaches, but our final E is the notion of bridgE, ie.  bridging from programming in Scratch to mathematics and vice versa. We also recognised the need to support teachers with the pedagogical approach through teacher professional development. We experienced what can happen if the teachers are not supported. Professional development was essential, but we also recognised that addressing a year's curriculum in two days of CPD, whilst also providing opportunities to engage with the second year of the 2 curriculum, was insurmountable.

My doctoral research focused on the teachers within the ScratchMaths project, and their developing mathematical knowledge as they learn to program. The complexity of identifying that knowledge is presented in Figure 10, an adaptation of Thomas and Palmer's (2014) Pedagogical Technological Knowledge for teaching mathematics (PTK).

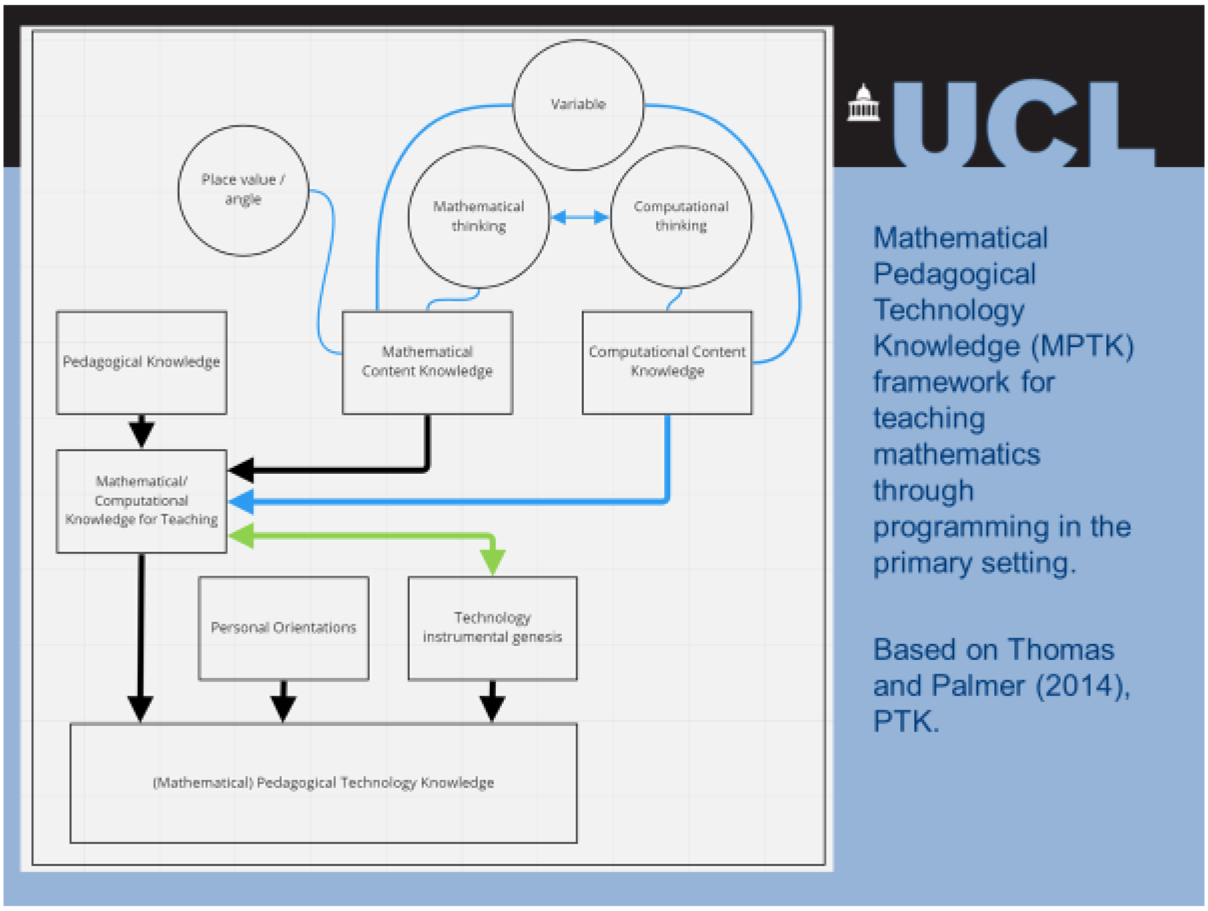
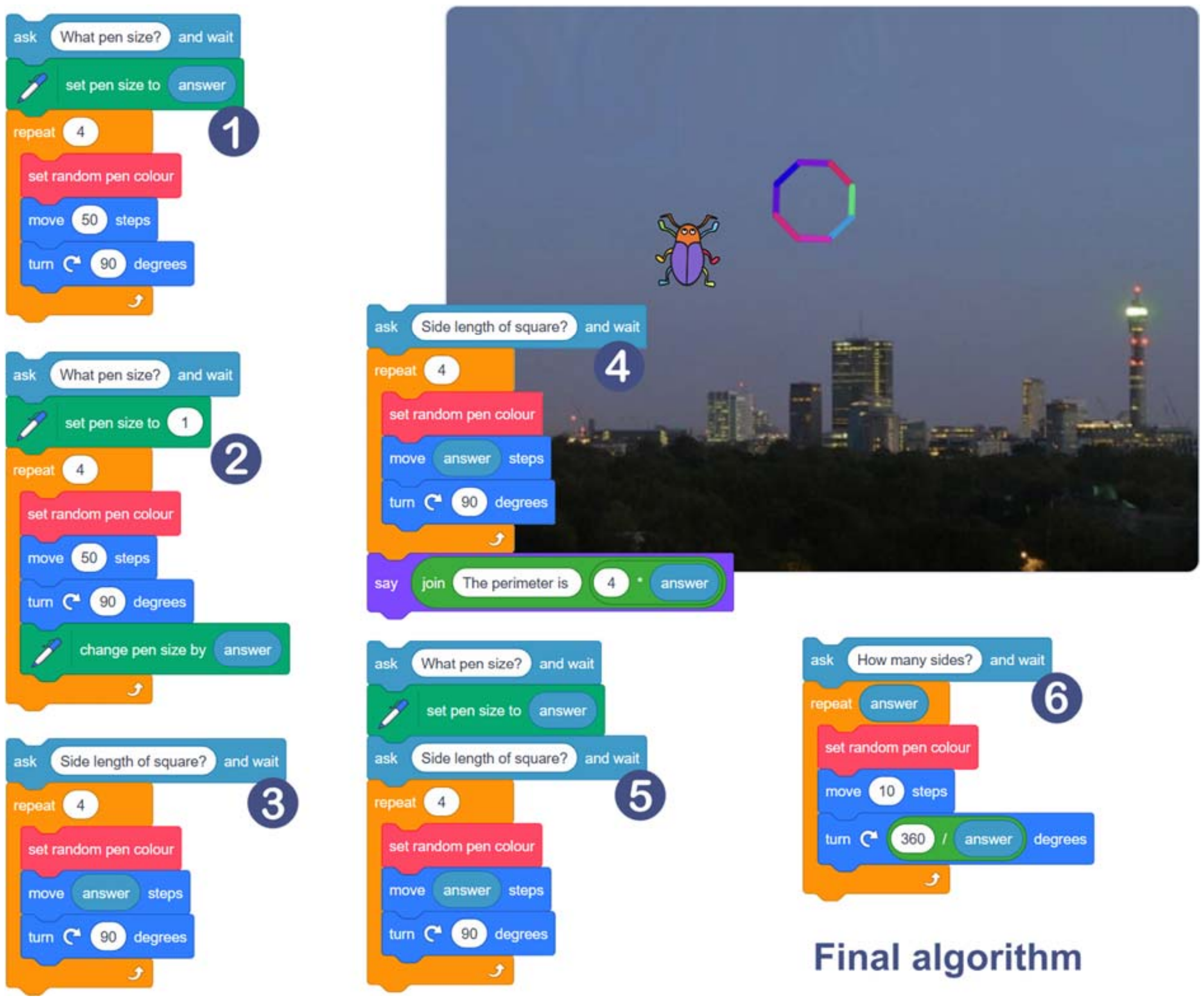

Variable
Place value / angle
Mathematical thinking
Computational thinking
Pedagogical Knowledge
Mathematical Content Knowledge
Computational Content Knowledge
Mathematical/ Computational Knowledge for Teaching
Personal Orientations
Technology instrumental genesis
(Mathematical) Pedagogical Technology Knowledge
UCL
Mathematical Pedagogical Technology Knowledge (MPTK) framework for teaching mathematics through programming in the primary setting.
Based on Thomas and Palmer (2014), PTK.

**Figure 10: Elaboration of Thomas & Palmer's MPTK Framework (Saunders, 2021)**

I add a box to denote Computational Content Knowledge, which feeds into the mathematical computational knowledge for teaching. I have added the mathematical ideas (in circles) which were the focus of both ScratchMaths and the specific focus of my doctoral research. For example, a variable has a very specific use in mathematics and also a very specific use in computing, but there are also overlaps, which were difficult for the teachers to articulate. This was challenging since the teachers felt they had a good understanding of algebraic thinking and that the concept of variable in computing would be the same. To illustrate this complexity, I use the example in Figure 11.



```
ask What pen size? and wait
set pen size to answer
repeat 4
    set random pen colour
    move 50 steps
    turn ↻ 90 degrees
```
1

```
ask What pen size? and wait
set pen size to 1
repeat 4
    set random pen colour
    move 50 steps
    turn ↻ 90 degrees
    change pen size by answer
```
2

```
ask Side length of square? and wait
repeat 4
    set random pen colour
    move answer steps
    turn ↻ 90 degrees
```
3

```
ask Side length of square? and wait
repeat 4
    set random pen colour
    move answer steps
    turn ↻ 90 degrees
say join The perimeter is (4 * answer)
```
4

```
ask What pen size? and wait
set pen size to answer
ask Side length of square? and wait
repeat 4
    set random pen colour
    move answer steps
    turn ↻ 90 degrees
```
5

```
ask How many sides? and wait
repeat answer
    set random pen colour
    move 10 steps
    turn ↻ (360 / answer) degrees
```
6

**Figure 11: The ScratchMaths Firework task**

On the right hand side of Figure 11 is a beautiful Scratch algorithm (Step 6). The algorithm **asks** how many sides, then stores the value in the **answer** block, and a regular Polygon will be drawn. The representation in programming exposes the mathematical structure, for example the relationship between 360 degrees and the number of sides. However, we respected the teachers' starting position so when we designed the guidance for this task, we supported a step by step approach, rather than just providing the final algorithm which brings together computational and mathematical ideas. To illustrate the approach we started with a simple algorithm to draw a square (see step 1 of Figure 11) which uses the **answer** block to store a variable and then use that **variable** to set the **pen size**. Step one therefore presents an **ask** and **answer** block consecutively, a relatively simplistic use of a variable in computing.

We then move to step 2, which uses the same **pen size** again, but now the **pen size** block is moving further from the repeat block. This incremental approach proceeds through Steps 3-5 until we arrive at the final algorithm, which is Step 6 on the right. This approach supported both teachers and their children to develop their programming skills and use those skills to explore mathematical ideas, in this example, the idea of a variable and the concept of external angle.

Moving to consider how local, regional or national initiatives are supporting teachers' professional learning needs in England, the context is set within the new national computing curriculum. It is a separate subject that has to be taught in all state-funded primary school by teachers who have trained as generalists, that is they teach every subject. This provided an opportunity to draw on the experiences of our team, who had worked with Logo and mathematics for nearly 30 years, and reconnect mathematics with computer programming, i.e., to exploit the synergy between the two. We recognised quickly that in order to work at a work at a national level, we needed to have regional support. Hence, we worked with, or created regional hubs where particular local agencies had a good knowledge of the schools, their context, and the types of technical infrastructure in the school.

My final point is perhaps obvious in that continued professional development is essential to sustain any new curricula, but it is so frequently overlooked. As I have discussed this is not just about the curriculum itself, but also about the specific pedagogy that is required to teach such a curriculum. Within our project, we had embedded professional development (PD) opportunities each year, but found very quickly that due to the high turnover of staff in schools, even this PD was not sufficient to support the teachers who had not taught the first year of the curriculum. This particular group of teachers required further support, which we had not envisaged at the start of the project.

### Response 2 by Iveta

What we know from our survey responses (Turgut et al., 2022), is that the most common obstacle for Norwegian mathematics teachers to implementing CTaP tools in mathematics teaching is their lack of competency and access to training. This is not surprising since most mathematics teachers were not educated to teach CT and/or programming. As a solution, the municipalities in Norway are offering professional development courses, which in Trondheim, involves a cooperation with NTNU.

Regarding teacher preparation, we find ourselves in an interim period whereby our student teachers have not experienced or learned any programming during their own primary or secondary education, and so are struggling. Thus, we have had to offer an "on the spot" elective course, on programming. In addition, as teacher educators, we also try to implement programming in our regular teaching. So we discuss with the student teachers how this might be accomplished in probability, in statistics, in geometry, etc, following the curriculum competency goals related to programming. I have asked colleagues from other universities about their experiences. In Oslo, for example, the situation is the same. We are all trying to adjust our courses accordingly, and although we may still be a little bewildered, we do our best!

### Response 2 by Ivan

In the ScratchMaths project Piers has just presented we worked closely with the in-service teachers. Such efforts are essential for successful transition. Iveta talked about how NTNU is trying to respond to the new situation as they support the preparation of their future teachers. This is also essential and crucial in the efforts of countries and institutions to bring about change, and I want to underline one thing. Unlike with mathematics, I believe, both in-service teachers and our university students, future teachers of informatics or mathematics, have most probably not experienced any programming during their primary and secondary school years at all. Almost certainly not in informatics classes and certainly not in mathematics. So if they come to our PD sessions, or if they are studying teacher programs, they have to learn both the actual subject content and the skills that a teacher needs for teaching it.

Along the lines of developing their digital literacy, the use of digital technologies is similarly challenging. It is crucial that future teachers experience productive use of digital technologies in all courses – whenever appropriate, during their university studies at the latest. Not just in a special course focused on the use of digital technologies for teaching/learning.

## Summary reaction by Alison

Thank you very much to the panel presenters, who have taken us on a complex journey through what is an immensely challenging topic. When we create the curriculum statements in national, regional and school documents, we assemble words on a page. We saw from Iveta’s research that a critical starting point is how teachers come to understand the meaning of these wolds. All teachers arrive with their different backgrounds and experiences. Some may have experience in computational work and programming, others may not. Some may be strong mathematically, others may not. So, building shared understandings of the teacher's perspectives on these words is critical as we plan how we can support teachers to implement the related curricula.

A second important group is the designers of the educational resources being developed to teach computational ideas within/alongside the teaching of mathematics. Both Ivan and Piers spoke very strongly to this. It is critical to understand the epistemological goals for both the mathematical curriculum ***and*** the computing, programming or algorithmics curriculum, which are influenced by the choice or design of technology.

I was impressed by Ivan’s introductory statement, which acknowledges the vast human global history of mathematics as an evolving discipline and a set of practices. It is a subject that is so established in our culture, and although we still struggle to understand many aspects of its teaching and learning, we have many years of human experience in terms of both mathematics and mathematics education.

Put alongside, computer education is relatively new to us. So, we rely on the computing experts to help us through. If we only had the expert mathematicians explaining and telling us how to approach the teaching of mathematics, we know that the mathematics educators would react – we have all heard of the “maths wars” that erupt around the world! Consequently, if the computing experts are working alone, there is the danger that the teaching of computing science in many countries will remain successful only for the minority of students who might elect to take such courses.

When we introduce aspects of computational thinking into a national curriculum, we have to really understand the curriculum goals at the highest level.

- Why do we think it's important that all children have some experience of learning ideas from computing?
- Why is it relevant for this learning to take place within mathematics?

Following on, what explicit design principles flow from these high-level goals? How do the educational content designers understand their design task(s), and its underlying rationale?

We heard from Ivan in this respect. He was very explicit about a design feature of the Emil technology for the kindergarten-aged children. It did not provide any feedback, an interesting point, which he justified in relation to the epistemological foundations of the design in Papert's constructionism. Conversely, within the Norwegian context, the technology was being selected due to its feedback function, interpreted by the notion of a computer programme "working" or not. These are contrasting approaches for the way that feedback is being considered in each context that underpin both the technology and task design decisions.

Piers and Ivan presented conceptions of pedagogical frameworks within the context of CT. This leads us to question, what do we mean by a pedagogical framework? and how might a well-defined pedagogical framework support designers to create resources that are going to be robust enough to keep the epistemic goals that underpin learning and teaching intact across a wide range of classroom contexts. This has a particular importance when the exciting (possibly high stakes) assessment processes may be misaligned with those epistemic goals.

Moving to assessment, which is a globally complex topic that is highly politically charged and, despite international rankings dominating the national and regional approaches, no one country in the world has developed approaches that embed formative, ipsative and summative assessment of CT. A starting point would be to research teachers' *in-the-classroom* strategies that help them to make sense of learners' progressions in their understanding of this new curriculum. Ivan's description of the way that primary teachers support learners in Slovakian classrooms implied a high level of teacher listening and whole class discussion.

In our ScratchMath project experience in England, it was really challenging for teachers who were new to the computing content **and** new to the 5Es pedagogical approach to try to understand how to best support their students. For example managing multiple screens in these environments can be really challenging. What screen is the teacher sharing? How is the teacher interacting with children as they are learning?

We have not had an in depth look at the nature of the assessment tasks that are being designed for CT, but Iveta's second example did imply one that was *product focused.* The students are expected to apply the ideas from computing to solve a problem of their own design. This is quite a different approach to the items posed in most current high stakes assessments around the world.

Finally, reflecting on the teachers' current perspectives, there does not appear to be a wide expertise in our teaching workforce, and even this panel, as the considered experts in the field, are challenged to design the teacher PD approach, the design of the curriculum and the design of the assessment. In England, the response during the ScratchMaths project;s design year was to work alongside a smaller group of teachers to codesign the solutions to some of those challenges.

Looking to Norway, which is beginning with the implementation of a new curriculum, whilst the initial work is to co-design curriculum resources, we might also approach the design of assessment tasks in a similar way. By activating teachers in this way, with associated processes of peer-review to iterate and improve both task and associated assessment items, we offer a scalable approach to professional learning at a time when there is limited expertise to offer more traditional cascade PD models.

Overall, what we have heard from Ivan, Iveta and Piers about the development of CT in their respective countries is that, despite the fact that some of these curricula began to be implemented nearly ten years ago, we are still very, very early in the journey to establish computing education as a domain of knowledge and practice. We shouldn't be surprised by that, given that computational ideas are relatively new in the history of humanity. Our call to action is to continue to have dialogues such as this one, and to work together to understand and share different approaches that fit within the institutional and cultural settings of our different countries.

## References

Blackwell, A. (2002). What is Programming? Proc. of *14th Workshop of the Psychology of Programming Interest Group*, pp. 204–218.

Boylan, M., Demack, S., Wolstenholme, C., Reidy, J., & Reaney-Wood, S. (2018). *ScratchMaths Evaluation report and executive summary.* https://educationendowmentfoundation.org.uk/public/files/Projects/Evaluation_Reports/ScratchMaths.pdf

Cansu, F. K., Cansu, S. K. (2019). An Overview of Computational Thinking. International Journal of Computer Science Education in Schools, 3(1), 17-30.

Grover, S., & Pea, R. (2013). Computational thinking in K-12: a review of the state of the field. *Educational Researcher, 42*(1), 38–43.

Directorate of Education (2019a). *Curriculum for Mathematics year 1–10.* https://data.udir.no/kl06/v201906/laereplaner-lk20/MAT01-05.pdf?lang=eng

Directorate of Education (2019b). *Curriculum for Mathematics vg1 theoretical. Mathematics T.* https://data.udir.no/kl06/v201906/laereplaner-lk20/MAT09-01.pdf?lang=eng

Directorate of Education (2019c). *Curriculum for Music.* https://data.udir.no/kl06/v201906/laereplaner-lk20/MUS01-02.pdf?lang=eng

Directorate of Education (2020). *Læreplan i matematikk for realfag (matematikk R).* [Mathematics curriculum for science subjects (mathematics R)] https://data.udir.no/kl06/v201906/laereplaner-lk20/MAT03-02.pdf?lang=nob

Directorate of Education (2022a). *Eksempeloppgave, MAT01-05 Matematikk, Del 2.* [Example set, MAT01-05 Mathematics, Part 2] https://matematikk.net/matteprat/download/file.php?id=3965

Directorate of Education (2022b). *Forslag til vurderingskriterier, MAT01-05 Matematikk.* [Proposal for assessment criteria, MAT01-05 Mathematics] https://www.udir.no/contentassets/a97119d8db0f476eb6f7e9d4adb51e41/losningsforslag_samlet_v2022.pdf

Gjøvik, Ø., & Torkildsen, H. A. (2019). Algoritmisk tenkning. [Computational thinking] *Tangenten–tidsskrift for matematikkundervisning, 30*(3), 31-37.

Grimsgaard, H.L. (2022). *Læreres profesjonsfaglige digitale kompetanse i programmering og algoritmisk tenkning.* [Teachers' professional digital competence in programming and computational thinking]. [Master's thesis, Western Norway University of Applied Sciences]. HVL Open. https://hvlopen.brage.unit.no/hvlopen-xmlui/handle/11250/3001030

Hopfenbeck, T. H., Throndsen, I. S., Lie, S., & Dale, E. L. (2012). Assessment with distinctly defined criteria: A research study of a national project. *Policy Futures in Education, 10*, 421–433.

Inprasitha, M. (2021). Preface. In M. Inprasitha, N. Changsri & N. Boonsena. (Eds.) *Proceedings of the 44th Conference of the International Group for the Psychology of Mathematics Education* (Vol.1). Khon Kaen, Thailand: PME.

Kalas, I., Benton, L. (2017). Defining procedures in early computing education. In: Tatnall, A., Webb, M. (eds.): Tomorrow's learning: Involving everyone. Learning with and about technologies and computing. *WCCE 2017. IFIP Advances in Information and Communication Technology*, Cham, Springer. Vol 515. pp. 567–578.

Kalas, I., Horvathova, K. (2022). Programming concepts in lower primary years and their cognitive demands. *Digital Transformation of Education and Learning – Past, Present and Future, OCCE 2021*. Cham, Springer Nature, 2022. pp. 28-40.

Kalas, I., Hrusecka, A. (2023). Levels of Control in Primary Robotics. Accepted for WCCE 2022 post-conference book, Cham, Springer.

Kveseth, S.S. (2022). *Programmeringsverktøy i matematikkundervisning.* [Programming tools in mathematics education]. [Master's thesis, Norwegian University of Science and Technology]. NTNU Open. https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/3007904

Nordby, S.K., Bjerke, A.H., & Mifsud, L. (2022). Primary Mathematics Teachers' Understanding of Computational Thinking. *Künstliche Intelligenz 36*, 35–46. https://doi.org/10.1007/s13218-021-00750-6

Nortvedt, G.A., Santos, L., & Pinto, J. (2016). Assessment for learning in Norway and Portugal: the case of primary school mathematics teaching, *Assessment in Education: Principles, Policy & Practice, 23*:3, 377–395, DOI: 10.1080/0969594X.2015.1108900

Noss, R., Hoyles, C., Saunders, P., Clark-Wilson, A., Benton, L., Kalas, I. (2020). Constructionism can work: the story of ScratchMaths. In Holbert, N., Berland, M., Kafai, Y. (Eds.), *Designing Constructionist Futures: The Art, Theory, and Practice of Learning Designs*. (pp. 39-52). Cambridge, Massachusetts: MIT Press.

Papert, S. (1980). Mindstorms. Children, Computers, and Powerful Ideas. Basic Books, New York.

Saunders, Piers; (2022) Tracing the evolution of teachers' mathematical knowledge and pedagogy through programming: Learning from Scratch. Doctoral thesis (Ph.D), UCL (University College London). https://discovery.ucl.ac.uk/id/eprint/10147323/

Sevik, K., & Guttormsgaard, V. (2019, May 8-10). *Algoritmisk tenkning og programmering i fagfornyelsen* [Computational thinking and programming in subject renewal]. [PowerPoint slides]. Nasjonal konferanse om bruk av IKT i utdanning og læring. Trondheim, Norway. https://www.nkul.no/wp-content/uploads/2019-S1G.pdf

Thomas, M. O. J., & Palmer, J. (2014). Teaching with digital technology: Obstacles and opportunities. In A. Clark-Wilson, O. Robutti, & N. Sinclair (Eds.), *The mathematics teacher in the digital era: An international perspective on technology focused professional development* (pp. 71–89). Springer.

Tofteberg, G. N., Tangen, J., Bråthe, L. T., Stedøy, I. & Alseth, B. (2020). *Maximum 8 Matematikk.* [Maximum 8 Mathematics]. Gyldendal Norsk Forslag.

Turgut, M., Gjøvik, Ø., & Kohanová, I. (2022). *Surveying Norwegian mathematics teachers' implementation and use of tools for computational thinking and programming.* [Manuscript in preparation]. Department of Teacher Education, NTNU in Trondheim.