Symposium Report: Report of Working Group on the integration of computational thinking and mathematics teaching and learning in Preschool to Undergraduate and Teacher Education Settings. In Online Proceedings of the *Computational Thinking in Mathematics Education Symposium*, UOIT, Oshawa (Canada) October 2017. Yiu et al.

REPORT OF WORKING GROUP ON THE INTEGRATION OF COMPUTATIONAL THINKING AND MATHEMATICS TEACHING AND LEARNING IN PRESCHOOL TO UNDERGRADUATE AND TEACHER EDUCATION SETTINGS

Steven Khan, *University of Alberta*
Immaculate Namukasa, *Western University*
Chris Yiu, *Western University*
Sheree Rodney, *SFU*
Victoria Guyevskey, *SFU*

WORKING GROUP PARTICIPANTS

Alison Macaulay (ON MoE - Curriculum),
Brandon Zoras (ON MoE - Curriculum)
Chris Yiu (UWO - Computer science, learning)
Immaculate Namukasa (Western University - Teacher education)
Sheree Rodney (SFU - Teacher education, elementary education, Dynamic Geometry)
Steven Khan (UAlberta - Teacher education, Parent education),
Victoria Guyevskey (SFU - Dynamic geometry, elementary education)

KEYWORDS: values, computational thinking, abstraction, automation, decomposition, debugging, algorithm, integration, mathematical thinking

ABSTRACT

In this working group, we explored the integration of computational thinking (CT) in mathematics teaching and learning from preschool to undergraduate and teacher education settings. We examine CT as an enigmatic term, with its' roots grounded in the work of Seymour Papert (1993, 1996) and focused on what it might look like and entail at various levels of education. In doing so, we foregrounded CT as a "value-driven" habit of human beings and examined a possible framework, within which CT is situated. We find resonance in the interdisciplinary nature of CT and suggest that for effective integration, there should be a focus on curriculum reconceptualization. In order to pursue this suggestion, we provide a small set of examples, which we believe promote CT in mathematics teaching and learning to show that, despite the use of CT as synonymous with computer programming, CT is more than programming and coding and is embedded in mathematics teaching and learning across all levels.

Symposium Report: Report of Working Group on the integration of computational thinking and mathematics teaching and learning in Preschool to Undergraduate and Teacher Education Settings. In Online Proceedings of the *Computational Thinking in Mathematics Education Symposium*, UOIT, Oshawa (Canada) October 2017. Yiu et al.

INTRODUCTION

We start from the premise that Computational Thinking (CT), like other patterns and categorised forms of thinking (Mathematical, Probabilistic, Design, Systems, Legal, Historical, Scenario, Evolutionary, Ethical, Critical etc.) is: a) thinking about some types of phenomena (objects, practices, principles) *in a particular* way that is related to, but still distinguishable, from other forms of thinking about the same or similar phenomena; b) situated within, and results from, the interactions of particular collections/collectives of individuals-with-tools-and-stories(i.e., interest-driven communities, involved in computational practices); and c) resulting in the emergence of a variety of computational cultures over time and geographical space. At present, there is a lack of consensus on definitions of computational thinking, and thus it remains a polysemous term that is often used interchangeably with Computer Science or Structured Computer Programming, in contexts in and out of education.

We understand this polysemy to be a consequence of the existence, contributions, and interests of multiple diverse computational communities in which the term has meaning and significance. Consequently, the attendant ambiguity for teachers and researchers surrounding the usage and definition of the term exists in educational settings, ranging from preschool through undergraduate and teacher education. Tatar et al. (2017), for example, note that in the North American context, there have been at least three foci to computational thinking at the school level: a direct approach through structured programming (e.g., Scratch, Python), abstract cognition (e.g. principles and concepts from Computer Science), and replicating actions that embody computational thinking concepts (e.g. robotics, unplugged activities). Similarly, in Mathematics Education, Harouni's (2015) genealogical analysis of why Americans teach the mathematics they do posits that at least three different institutional settings - the Shop floor, the Grammar school and Reckoning schools and their associated cultures and values, as bequeathing, to varying degrees, their genetic legacies to modern mathematics education. Harouni concludes that the curriculum and pedagogy of today's American mathematics education is most aligned with a hybrid of Reckoning and Grammar school curriculum and pedagogy. Understanding the complex genealogy of the concepts we work with, the socio-political contexts in which they operate and their historical alignments with power and privilege is a critical requirement in appreciating and addressing the challenge of curriculum integration involving CT.

Tedre and Denning (2016) provide a brief and necessary history of the development, diversification, and mythologisation of the concept of CT over more than half a century, beginning in the 1950's. They demonstrate that, "the key concepts, narratives, and major arguments of CT were worked out during many years of debate from the 1950s to the 1990s [with]...definitions var[ying] from narrow to broad" (p.127). Recognition and awareness of the breadth and depth of this history is important for work in education to reduce the risk of, "repeating already refuted claims, past mistakes and already solved problems, or losing some of the richest and most ambitious ideas in CT" (p.120). They note that the various computational sciences (eg. bioinformatics, computational fluid dynamics etc.) in existence today each have their preferred and dominant computational models, and stress that CT is always in reference to a specific and implied model. To summarize, different computational cultures, their dominant computational models, metaphors and practices have contributed and contribute to the plurality of approaches to defining CT in the field today.

Our working group engaged in several discussions over the two-day period, as we explored the integration of computational thinking (CT) in mathematics teaching and learning from preschool to undergraduate and teacher education settings. We first present our discussion regarding a working definition of CT, and then its relationship to mathematical thinking. In the next section, we explore CT as a value-driven process. We present several examples we considered as we worked through these discussions. We end with a consideration of the need for curriculum reconceptualization in order to better pursue CT as a goal in an integrated framework and present the idea of proto-computational thinking (PCT) as a slight, but perhaps meaningful, shift in the conversation.

DISCUSSION: WHAT IS COMPUTATIONAL THINKING?

A significant part of our discussion on both days involved a consideration of the multiplicity of definitions of CT, of which we were aware at the time through works such as Denning (2016); Weintrop et al (2016) and Wing (2006). This discussion provided a common framework for productive discussion and engagement with the idea of curriculum integration. We note, from previous experiences in symposia and working groups that this is often a necessary first step, due to the diversity of experiences and conceptions of CT (Buteau et al. 2016; Namukasa et al., 2015).

In education, we noted that Seymour Papert (1993, 1996) is credited with introducing the CT term, and Jeanette Wing (2006) is credited with popularizing it as a concept. Shute et al. (2017) however, after an extensive literature review suggests a definition of CT as "the *conceptual foundation* required to solve problems effectively and efficiently (i.e. algorithmically, with or without the assistance of computers) with *solutions that are reusable* in different contexts" (p.142, italics added), which is an elaboration of the Cuny-Snyder-Wing definition of CT as "the thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent" (2010). The idea of reusability in different contexts as they related to abstraction and automation were themes that we returned to and are discussed in the section below on CT as a value-driven human. Our discussion also touched on the fact that in CT understanding is a goal and that for CT one is actively understanding and thinking about the situation and goal.

Simultaneously, we considered and tried to represent the relationship between CT and Mathematical Thinking (MT), using Venn Diagrams. Our eventual preference was for an intersecting sets representation (Fig.1), which captured our belief that there are elements in common and there are elements that are distinct, rather than one being completely a subset of the other, or the two either being completely unrelated, or fully overlapping.

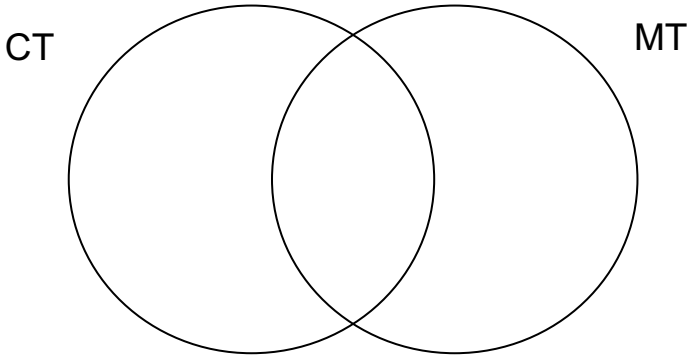


Figure 1: Perceived relationship between CT and MT.

The question of how to introduce the idea of CT to children and young adults within mathematics education without an agreed-upon definition in the field was another recurrent theme in our discussions. As an anchor to our discussion, we used Noss and Hoyles' framework, presented at the 2015 symposium, which defined CT as involving Pattern Recognition, Decomposition, Abstraction, and Algorithm Design. Pattern recognition is seeing the data of the problem and understanding the data and variables; decomposition is looking at a problem, really understanding it, breaking it down into pieces, analysing its' relationship with previously encountered models and patterns. Also included the ability to recognize a problem by its *type* in order to identify parts that can be re-used or remixed. We felt that what separates CT from other forms of mathematical problem solving is extracting/abstracting the answer to the problem into other domains, and to apply the solution to other cases. We also noted that CT is a process that you follow to solve a problem. Namukasa, Minakshi & Miller (2017) suggest this close relationship with the metaphor of problem solving as used in mathematics education:

> The metaphor of CT as extending over the problem-solving process, which is also present in Wing's (2006, 2011) definition of CT, resonates with our interest in integrating CT in school curricular. For Kalelioglu, Gülbahar and Kukul (2016) CT is about the whole process of problem solving including: identifying a problem (through abstraction, decomposition, levels thinking etc.), working with information and data needed (data practices, pattern recognition, conceptualizing etc.), planning solutions (logic, algorithms, procedures, parallelization), implementing solutions (automation, modelling, simulation) and assessing solutions (testing, debugging, generalization) for further improvement. Other metaphors such as designing computational projects, expression through a variety of computational media, computational model building are also applicable to CT.

There was a suggested link to Polya's (1945) problem-solving model with CT, using the metaphor of problem solving: (1) identify problem through decomposition and abstraction (Polya's Understanding phase), (2) working with information and data needed (through data practices, pattern recognition, conceptualizing), (3) planning solutions through logic (algorithms), (4) implementation (through automation, modeling, simulation), and (5) assessing the solution (through testing, debugging, for further improvement). Tedre and Denning (2016) note that historically (1970), Feurzig, Papert and colleagues had explicitly argued that, "programming was a great tool for concretizing Polya's classic text

How to Solve It on problem solving in mathematics" (p.122), and that amongst his group, there was an evolution from "mathematical *rigorous thinking*, to *procedural thinking*, and *computational thinking*" (p.122). This reinforces the need for those of us in the field of mathematics education, who look to integrate math and CT, to become familiar with the history and trajectory of the disciplinary fields.

Recommendations: Diverse intersectional groups, such as ours, who work at the nexus of mathematics education, computational sciences, teacher education, and PreK-16 curriculum, need to have a greater awareness of the history of the fields and the diverse cultures, concepts, and individuals that have contributed to and helped to formulate these fields. This does not resolve the definitional problem of what CT or MT are in a rigid ontological sense but allows us to make sense of the diversity in a, hopefully, more productive way. We noted that consequently, there is a need, perhaps even an ethical requirement, for those of us doing work with diverse publics and in knowledge mobilization efforts, to be more precise with our language, and reserved with our claims in our research reports and in public communications about CT. We do need to take care and time to explicitly identify the specific aspect(s) of CT that we are addressing, so that the part or aspect of CT is not mistaken for the whole. We do not want to create the conditions for misconceptions or myths to arise; such as the notion that CT equals programming or coding, or that CT skills have far-reaching transfer - both of which are not supported in the research literature (Tedre & Denning, 2016).

DISCUSSION: CT AS A VALUE-DRIVEN PROCESS

Foregrounding the socio-cultural dimension, we presented earlier, we have come to believe that it is important for us as researchers, teachers, and educators to think of CT, and explicitly name it as a **value-driven habit** of human-beings. We are not aware at this kind of research on CT in non-human animals or other species at this time that might refute this claim. We do acknowledge evidence for computational principles in life processes (Flake, 1998; Pavlus, 2016) but draw the line for now at the 'thinking' aspect. That is, we see CT as an enculturated set of human practices to see, hear, encounter, and ultimately read and write the world, in a Freirian sense, in **particular** ways that are valued/rewarded in specific computational cultures. Our challenge, similar to the problem of definitional consensus, was agreeing on what those particular ways might be for us, whose work intersects with education. In focusing on what values drive CT habits of mind and forms of practice, we note that this dimension has both context-dependent features and subjective features. The ideas of **Abstraction and Automation,** we believe, lie at the intersection of the two sets above, i.e. it is part of the value system of both the Mathematical Sciences and the Computational Sciences, and core to both MT and CT.

We saw these two principles or processes as intimately related in the following way: abstraction enables and makes possible automation (often through pattern recognition and cross-connection processes). In the specific context of various computational sciences and cultures this is frequently through the design of algorithms. In our discussion, we suggested that what separates CT from other more traditional problem solving is this explicit value and intention to extract/abstract those elements of an answer to a specific bounded problem, to transfer those aspect that are amenable to implementation via algorithm and to apply the solution to other cases. It may also bring to bear specific tools and concepts from the computational sciences to a more traditional mathematically posed problem, for example, the case of the solution of the Kadison-Singer problem (see Klarreich, 2015). We tentatively framed this using Hewitt's (1999) distinctions between necessary and arbitrary aspects, though here in relation to elements of the problem that could be algorithmised (necessary), and the elements of the problem that were specific and substitutable (arbitrary). We used the example of the owl problem (see Section on Examples) to illustrate this. We note as well that among Gadanidis' (2017) set of five affordances of CT for elementary mathematics education, Abstraction and Automation are included. The other three values-drivers - agency, access and audience (Gadanidis, 2017) - we believe, are emergent from and situated within educational and school contexts through of relevance to other computational cultures more broadly. A route for further discussion in the future is the implications of this abstraction and automation for our local and global computational cultures. This was outside the scope of the working group focus.

In pursuing these ideas after the symposium working group, we have come across an online essay by Martinho-Truswell (2018), in which he proposes the criterion of automation as a defining human characteristic, with our tools and sequenced connections of tools (a 'machine' in the Deleuzian sense of a set of connections, not a set of parts) increasingly allowing the offloading of physical labour, and in some societies - cognitive labour. This corresponds well with views of embodied cognition and enactivist thought in mathematics education. This "cognitive automation", Martinho-Truswell suggests, is the result of the need to overcome the limitations of working as well as long-term memory, especially in situations of increasing socio-economic complexity, where 'trust' is essential. The invention of the Jacquard loom in the 19th century, he suggests, recombined cognitive and physical automation in ways that threads their way into our modern digital devices. All of our computational activities, he offers, are in the service of human goals, even if they are but "layers of human instructions committed to external memory being carried out by machines that can read it" (online, n.p.).

Although Martinho-Truswell describes the drive as exporting effortful tasks, it is perhaps better to say that tasks, which are automated through CT and other human habits of mind, are those that are effortful (physically, cognitively or both), dangerous, humanly impossible, or perceived as joyless (repetitive), but at the same time, confer some

competitive advantage (often economic) to individuals or organizations. This view of humans as apes-who-automate, allowing us "to supplement our bodies and brains with layer upon layer of external assistance [giving]...a depth, breadth and permanence of mental and physical capability that no other animal approaches" (online, n.p.) is perhaps another way to explain what CT is, how it fits with who we are as a species, and why it is aligned with curriculum.

RECOMMENDATIONS

In future discussion, we may wish to attend in a focused way to the broader social, ethical, and curricular implications of abstraction and automation, as well as other values that integrating CT with school curriculum have to offer.

Exploration: Examples and non-examples of CT

Alongside the discussions presented above we brought in several examples over the two days to help us try to make sense of what CT is and what values drive CT habits and practices. In the report, we have collected the examples (including CT in problem solving contexts as well as in non-programming and programming contexts) here in one section with notes on the discussion related to each. We were also considering how the tasks would relate to curriculum PreK-16 and in teacher education. We looked for interesting examples from our own experience that would allow us to bring the distinctions between mathematical thinking and computational thinking to the foreground so that it might be clearer to us. We pondered 'What would be considered mathematics thinking and tools in a task as opposed to computation thinking and tools in these examples?' We also sought to identify and name certain periods of our thinking as mathematical or computational.

The Owl Problem

The two problems below appeared in the Global Math Challenge (website), an online math competition, in 2016 and are related tasks involving spatial reasoning. We considered how we might find and adapt tasks such as these to bring out some ideas in CT through changing the focus of our traditional questioning. We did note that solving the initial task does require spatial reasoning skills and that attempting to formulate that thinking algorithmically is a challenge - but this is the area of productive struggling and computational challenge.

Traditional Question requiring Mathematical/Spatial Thinking: Some owls are sitting on some blocks. The picture shows the front and back view of the same set of blocks. How many owls are there? Explain how you solved the problem using models, pictures, and/or words.

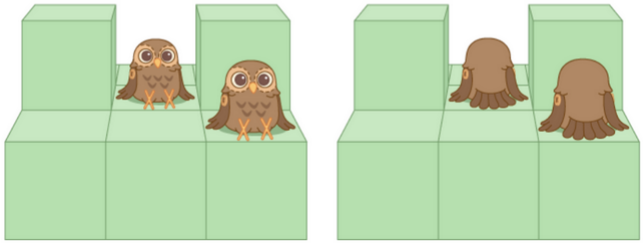


Figure 2: Owl Problem primary grades.

Questions to draw out CT processes: Write an explicit algorithm to solve problems like this based on your solution strategy Is your thinking different when you are writing the algorithm from when you were solving the problem? If so how? Test your algorithm on the following problem and make any changes. Give your algorithm to another person/group. Use the algorithm you have received EXACTLY as it is written/presented/defined. Do you get the same answer? If not is there something you learned? What is similar about the different algorithms and what is different? Create a different but similar problem to try to 'break' each other's algorithms, i.e. can you create a set-up or find ways of conditions where the algorithm won't be successful? Revise your own algorithm based on your thinking and finding. Give it so someone else to test. [We note as well that these problems could be 'gamified' so that a larger example space is explored before the questions are posed.]

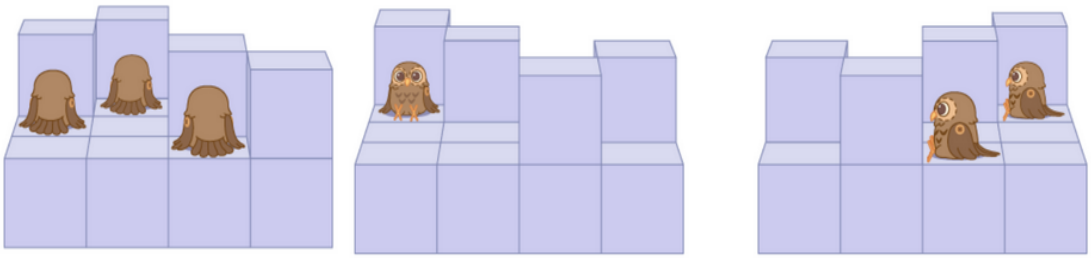


Figure 3: Owl Problem Junior Grades

In the Owl Problem, we saw the owls and their locations as arbitrary. Necessary aspects include the realization that the view of owls could be blocked depending on the position of the viewer and where other blocks are placed. We also noted that we have assumed that the blocks are solid, rather than just a folded surface and that there always exist blocks under those blocks that create a second level. In discussing algorithmic design to solve problems of this type, the abstraction of the relevant idea is that creating an accurate 'top view' of the system always allows us to count the number of owls directly even if they are behind other blocks. This strategy or algorithm generalizes to all problems of the type, "looking from the top-down can let us see what is behind obstacles in other views." The sequence of questions and the task also allows learners to question their assumptions and try to iteratively improve their solution so that it might work in a greater number of situations e.g. if some of our assumptions are not true. It also engages a 'hacker' mindset - to understand a system as deeply as possible.

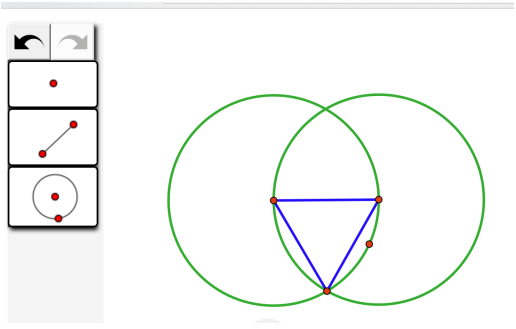
The task also provided us an opportunity to analyze our thinking and whether we could identify *when* we were thinking mathematically and when we were thinking computationally. We did note that individually our thinking is influenced by our past experiences and for most of us in the working group this has been mostly mathematical rather than computational. However, we wondered what if we analysed our experiences explicitly to look for CT ideas? What would we see? We note that it is a challenge to think about and look back on the thinking while working on tasks and to attend to where our awareness is and what it is attending to. We think this is an area for additional exploration.

Coding in a Geometric Environment (Web Sketchpad)

http://www.sfu.ca/geometry4yl/websketchpad.html.

In addition to using alphanumeric language, there are also "spatial programming" languages in which the input and output are both geometric (using geometric primitives such as circles, segments, and points). Sinclair and Patterson (2018) have shown how dynamic geometry environment (DGE) such as Sketchpad can function as spatial programming languages that also support mathematics learning.

For example, construction tasks in a DGE can be seen as involving programming. Points, segments and circles can be used to construct an equilateral triangle (Fig.4). The construction sequence functions as a procedure, which can be tested by dragging the independent objects in the sketch and checking whether the triangle remains equilateral. If the triangle falls apart or changes its properties, debugging (to correct a procedure) can be used to change the construction. This task could be extended to creating a regular hexagon (Fig. 5), in which case the same procedure needs to be repeated six times (to iterate a procedure).



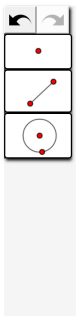


Figure 4: Equilateral triangle in Sketchpad

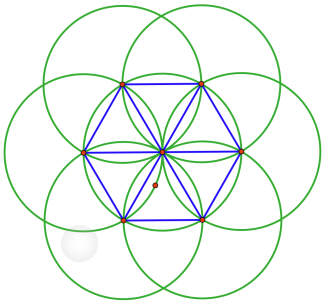


Figure 5: Regular hexagon in Sketchpad

How many solutions?

This problem is attributed to an April 2017 tweet by Graham Fletcher

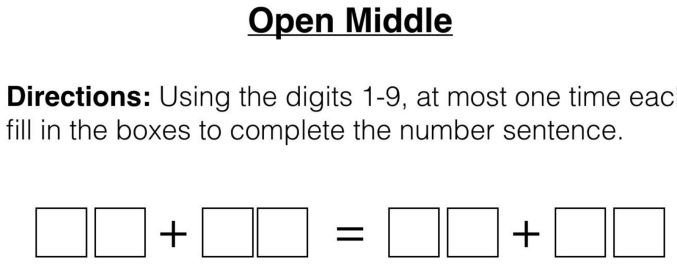


Figure 6: Open Middle problem (https://twitter.com/gfletchy/status/852494584538181634)

Although we did not engage with this significantly during the symposium working group the problem provides a good opportunity to compare mathematical and computational approaches and perhaps mathematical and computational thinking. The first mathematical approach is to try something and look for patterns and then generalize ideas like interchanging units digits or tens digits on one side of the equation to generate additional solutions. Steven had also posed the problem to Cameron Morland who is in computer science at University of Waterloo during CMESG 2017. He sent two python programs that arrive at the solutions differently - one using a dictionary: https://drive.google.com/file/d/1T2i24cjQjLawtfsiZyeo27QXoITcjeAB/view?usp=sharing and one using permutations: https://drive.google.com/file/d/17DXC3GO0-MwS7O8r_k-Be0Xgsk3SdXnD/view?usp=sharing.

The first solution, using a dictionary, calculates all of the sums of each pair of number from 10 to 99 inclusive (e.g. 10 + 11 = 21, 10+12 = 22, etc.). It then goes through all of the sums and checks every two pairs of numbers to ensure that every digit is unique from

1-9, and that there are no zeros in any numbers.

The second solution calculates all the different permutations from the digits 1 through 9, then takes the first eight digits of each permutation and checks if the sum of the first two pairs of digits equals the second two pairs of digits, that each digit is unique from the range of 1-9 inclusive, and that there are no zeros in any number.

Both solutions presented would generally be considered 'brute force' approaches. They take advantage of the fact that a computer can perform these calculations very quickly and do not attempt to use any domain knowledge to reduce the search space. The interesting part about these solutions is that they do not require any mathematical insight on the structure of the problem itself. This is not to say that these solutions do not require any mathematical thinking - understanding the problem and its search space requires mathematical knowledge especially with regards to bases, while programming the solution requires knowledge of permutations and modulus. However, this still demonstrates how a CT approach can work on a problem.

We do think that some problems that teachers might be utilizing for other mathematical purposes could be good test cases for elaborating and exploring the distinctions and affordances of computational thinking.

Olympiads in Informatics

Within mathematics education, competitions form a powerful sub-culture. There exist similar sub-cultures for computational thinking and computational practices. In both cases, there are often an explicit intent to identify 'talent' and we wondered about the values driving the different competition cultures. We used the problem below to question whether it could be used to identify students who have a propensity for CT or coding and to discuss the following belief: that it is possible to be good at CT and not be a good coder but that the reverse is not likely, given that coding provides a strong environment for developing CT proficiency.

Australian Informatics Competition : Sample questions, set 2, with solutions 4

4. Robot *2006 J.5*

The weather is lovely, so — like any other sensible person — you decide to spend a day on the beach with your pet robot.

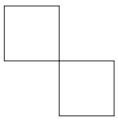
Your robot is fairly simple; all it can do is walk around the beach and trace patterns in the sand. It begins facing north and accepts the following instructions:

- `F`*x* : Walk forwards *x* centimetres.
- `L`*x* : Stay in the same place and turn *x* degrees to the left.
- `R`*n*`[...]` : Repeat whatever appears in the brackets *n* times.

For example, the robot can trace out an equilateral triangle in the sand using the instructions `F30 L120 F30 L120 F30 L120` (see the illustration below). This same pattern can be simplified by writing `R3 [ F30 L120 ]`, since the forward/left instructions are repeated three times.

Note that if you repeat the forward/left instructions four times instead of three, the pattern in the sand would be exactly the same (the robot simply walks over one edge of the triangle twice).

You are feeling artistic, and you would like the robot to trace out the pattern illustrated below.



You have given the robot the following instructions:

```
R2 [ R4 [ F50 L90 ] L90 ]
```

Unfortunately it does not trace out the pattern that you wanted. With a slap of the forehead, you realise that the number in one of your instructions was wrong. Which instruction had the incorrect number?

Figure 7: Olympiads in Informatics http://www.amt.edu.au/pdf/AICsolutionssampleset2.pdf

We wondered if it was possible to solve this problem using mathematical thinking or using CT. How this could be CT? An example is given (the triangle), in order to solve this and if you have no experience with programming and robots you have to be able to understand the pattern of how those instructions cause the robot to make those patterns in the first place. If you can't understand the pattern then you have no hope of solving the problem. Thus, the solver must abstract the solution to the problem. We felt that this type of problem might be a diagnostic type task for certain types of computational thinking. We moved on to considering if these are useful problems for school curriculum. We also felt this problem might be good as an exit task or a check for understanding.

A learning trajectories approach

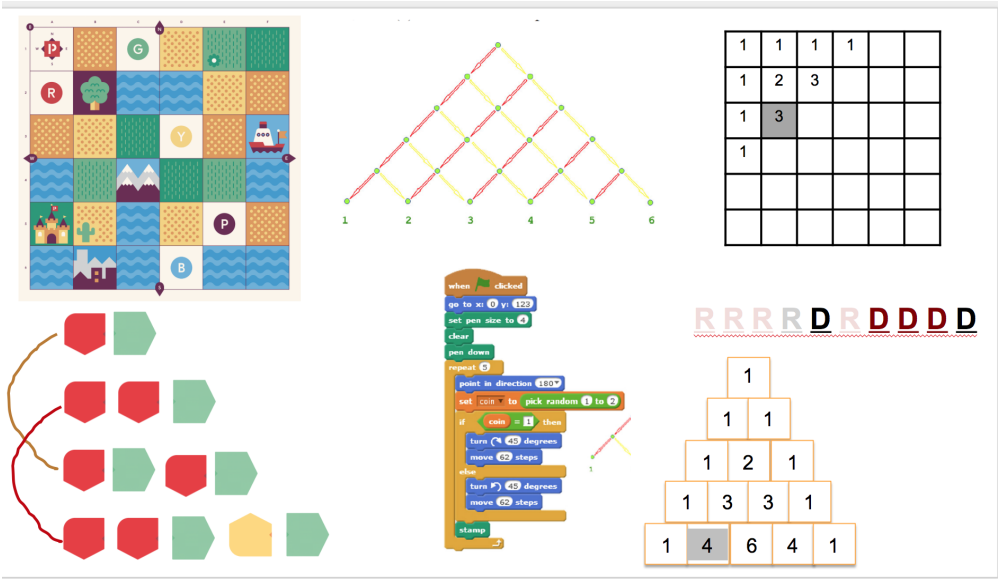
Learning trajectories have been receiving a lot of attention in the literature on teaching mathematics in the early years and we wondered if this lens might be of benefit in curriculum considerations. According to Clements and Sarama (2010), "learning trajectories have three parts: a) a mathematical goal; b) a developmental path along which children

develop to reach that goal; and c) a set of instructional activities, or tasks, matched to each of the levels of thinking in that path that help children develop higher levels of thinking" (online). With this in mind, we asked, 'What knowledge and resources do teachers need to have in their practice?' We noted that many were doing good things with coding. We felt that to achieve proficiency with any type of thinking one is required to practice with well thought out sequences of variation (see the example sample Space) that would allow them to develop the necessary skill in abstraction. We noted that in many observations we were seeing the rich experience and the hard fun, but were wondering if the necessary variation in experiences for the abstraction aspect was being provided, not only in our practice but in teachers' classrooms?

We shared Carly Rozin's (2016) work that she had presented at Fields around what makes for a good CT task. Rozin (2016) believes a good CT task has the following components:

> They are **modular** (can be broken into components / smaller pieces).
> Solving **smaller pieces** allows you to solve the original question.
> Understanding of the components leads to a **deeper understanding**.
> The deeper understanding allows you to build on the original solution or answer **more complicated questions**.

In Figure 8 below we juxtaposed the mathematical task of determining the number of paths on an nxn square grid from one corner to the one diagonally opposite where moves are only permitted downwards or to the right (far-right) with a coin flipping task coded in Scratch (middle) and the problem of navigating in Cubetto's world with physical coding blocks. We note that teachers (and learners) in Pre-K, early primary and Junior-Intermediate may encounter each problem as a discrete case without appreciating how they might be connected.



Cubetto's World **Coin Flip problem in Scratch** **How many paths?**

Figure 8: Three related problems at developmentally diverse ages?

RECOMMENDATIONS

We need additional examples and a clear hypothetical learning trajectory for how examples could be sequenced developmentally. There also needs to make clearer connections to curricula and clarity about what aspect of CT is being assessed and how it is being assessed. This relates back to our first recommendation regarding being clear and explicit about which aspect of CT we are focusing on in a given task so that it might be clear to teachers and students. We would like to see more explicit examples and guidance for teachers and curriculum developers where a well-known mathematical activity or task is used to bring out or develop specific elements of CT ideas and practices.

RECONCEPTUALIZING CT FOR THE CANADIAN CLASSROOM: INTERDISCIPLINARY LEARNING

In exploring a working definition, we also looked at efforts to integrate CT tasks in Canadian curriculum. Despite the high levels of interest in developing CT skills among 21st-century children, there have been challenges to address and identify the main characteristics of CT. Likewise, we heard that teachers who are integrating CT in teaching in different jurisdictions were unclear on ways how to integrate CT in the mathematics curriculum.

Wing's (2011) definition has provided two valuable viewpoints about CT. Firstly, CT is presented as a thought process, which makes it independent of technology, and secondly, CT is an explicit type of problem solving, which involves diverse abilities, such as being able to design solutions that can be performed by a computer, human, or by both. Our representation of the relationship between CT and MT is best represented by Venn Diagram for intersecting sets, with the ideas of abstraction and automation lying at the overlapping sections of the two, meaning that they are integral to both MT and CT. Gadanidis (2017) points out that the trend of adding some form of computer coding to curriculum is an international phenomenon, citing Kotsopoulos et al. (2017), who emphasize that England, Finland, Estonia, and USA have all mandated CT curriculum. He also highlights the benefits of integrating CT and mathematics in school:

> At the heart of computational thinking – and mathematics – is abstraction. When children write code, they come to (1) understand in a tangible way the abstractions that lie at the heart of mathematics, (2) dynamically model mathematics concepts and relationships, and (3) gain confidence in their own ability and agency as mathematics learners" (p.1).

It is our belief that CT is distinctly embedded in mathematics teaching and learning, and should be taught within - rather than alongside - mathematics curriculum. However, in order to achieve effective interdisciplinary integration, there should be a focus on curriculum reconceptualisation.

In 2017, Global News Network issued a detailed report on how coding is taught in Canadian schools, and summed it up as follows: Nova Scotia, New Brunswick, and British Columbia all made coding mandatory, with Ontario and Saskatchewan having it as an optional part of the curriculum, while Alberta, and Manitoba are looking at their options and "studying the approach taken in other provinces."

In BC's new curriculum, which was officially introduced in September 2016 in all K-9 classrooms, a subject of Applied Design, Skills, and Technologies has been added (BC Ministry of Education, 2017). Similarly, as of 2016, Nova Scotia has started Information and Communication Technology and Coding curriculum integration in both primary and intermediate grades. What is remarkable about the way the Applied Designs, Skills, and Technologies curriculum is introduced in BC- and a shift away from the previous autonomous model - is that there is an expectation for K-5 levels that the content will be taught "in combination with grade-level content from other areas of learning in cross-curricular activities to develop foundational mindsets and skills in design thinking and making." Middle school grades from 6 to 8 have an explicit Computational Thinking module outlined in the Learning Standards section. Students in grade 6 and 7 are expected to know simple algorithms that reflect computational thinking, visual representations of problems and data, evolution of programming languages, and visual programming - all taught in conjunction with curricular competencies of Applied Design, Defining, Ideating, Prototyping, Testing, Making, Sharing, and Applied Skills. In grade 8, more content is added and students are expected to know the following:

- software programs as specific and sequential instructions with algorithms that can be reliably repeated by others
- debugging algorithms and programs by breaking problems down into a series of sub-problems
- binary number system (1s and 0s) to represent data
- programming languages, including visual programming (Scratch, Alice, Greenfoot, BlueJ), in relation to text-based programming (HTML) and programming modular components (Arduino, LEGO, Mindstorms)

The five-year International Baccalaureate Middle Years Program (Interdisciplinary MYP Guide, IBO, 2014), designed for grades 6-10, outlines interdisciplinary integration as one of its requirements. Interdisciplinary learning, which is described as a process by which students integrate two or more disciplines to create new understanding, has three main characteristics: it's purposeful (not a goal in itself), grounded in disciplines (reorganizes disciplinary objectives in meaningful ways), and integrative (elements of one or more disciplines are placed in productive relationships with one another). These characteristics

align with our philosophy of curriculum reconceptualisation, and could be used as a guide to help successfully integrate CT and Mathematics curricula.

Mueller (et al, 2017) examine Ontario elementary curricula for CT concepts and skills using frequency and content analysis based on 38 keywords (eg. Algorithm. Data, Recursive, Simulate). They do not find the specific term in any curriculum document but note "an emphasis on CT-related concepts and perspectives in seemingly unrelated disciplines of language and the arts…" (p.265) in addition to alignment with mathematics and science curriculum through problem-solving. They note that specific terms related to computer programming cannot be found and suggest that, while concepts and perspectives foundational to CT are evident, the actual *practices* involved in CT are not named.

The existence of foundational concepts and perspectives of CT aligns with Tartar et al.'s (2017) findings in their study of co-developing integrated computational thinking tasks in Grades 6-12 with teachers. They note that, "the result of student classroom observations, interviews with teachers, and participatory engagement rarely led to activities that fit the prototype of CT if CT means using elements easily recognizable to computer scientists as computer science…" (p.65). Instead they found what they have labeled *proto*-computational thinking (PCT) which,

> consists of aspects of thought that may not put all the elements of CT together in a way that clearly distinguishes them from other human intellectual activity. PCT activities may draw out the use of a representation, stimulate thought about the utility of different representations, partially instantiate abstractions, or increase focus on a systems' level analysis. They might emphasize systematicity in problem solving but not emphasize problem solving in ways that are enactable on a computer" (pp. 65-66).

Returning to our socio-political framing of CT as a value-driven process, they go on to argue that, "PCT is not in competition with CT. But it is also not just a primitive kind of CT. It is a cultural phenomenon. The definition of CT, the goals of ICT, and the acknowledgment or failure to acknowledge the importance of PCT are political as well as epistemological acts and hence important because our well-meant interventions in the area of CT may reinforce existing social inequities" (p. 79). They found that in their study they were more successful at promoting PCT by promoting systematicity and a *need* for a systematic approach. This is an important point for our consideration - perhaps what we are doing at the elementary level is better termed developing foundational aspects of CT or proto-computational thinking (PCT)? Rather than trying to shoehorn a concept that is not well-fitted to curricula, and is not integrated, and which professionals in the computational disciplines might understand differently. This slight shift may open the conversation up in productive ways for researchers and teachers and curriculum designers.

RECOMMENDATIONS

The ongoing challenge to integration seems to be in those situations where CT is trying to be 'fit' into curricula as yet another 'add-on'. We see this approach as having limited viability over time. We acknowledge that teachers both pre-service and in-service require support from academic institutions, their local bodies and ministry offices. We believe the best chance for long term success is for CT to be integrated but that this requires a reconceptualization or at least a re-examination of the role and nature of curriculum documents, expectations and resources. We find the concept of Proto-Computational Thinking (PCT) very promising and view it as a more realistic and achievable systemic goal for naming what we do at the earliest grade levels. This keeps CT intellectually honest and respectful of the multiple responsibilities around competencies that teachers are already charged with developing as well as acknowledges the polysemy and ambitious challenge of integrating CT across the PreK-16 landscape.

CONCLUSION

Computational Thinking can be integrated but this requires a careful reconceptualization of curriculum. The current approach of bringing it into curriculum via mathematics is promising but additional examples and a developmental framework including tasks such as that of learning trajectories may be useful. Adapting existing mathematical tasks to explicitly draw attention and bring out computational ideas from a variety of sources is one avenue for additional exploration. The roles of abstraction and automation, and the implications of these need greater attention. We are unlikely to resolve the ontological question of what CT is in a way that is satisfactory to all the communities who use and have contributed to the term. However, we in educational research should be explicit, and perhaps more circumscribed in our communications to clearly name which aspect(s) of CT we are focusing on. There may also be some value in considering the concept of PCT for the early grades as a more accurate descriptor of what we hope to accomplish. We note that certain teachers are innovating in the ways they are integrating coding in their teaching of school curricula and so are really doing good things with coding, but we also want them to get really good at the thinking which requires repeated practice with well thought-out task sequences that vary in ways that allow teachers and students to engage in the thinking opportunities evoked by the tasks as well as in making some abstractions from the tasks. In many ways, this is analogous to what goes on for learners in constructionist environments such as Scratch, though there is a sense of a gap for those working towards developing CT within current curriculum and preparation structures. We noted that when you begin to learn computer science, you learn the tools - loops, data structures, variables, etc. and how they fit together. Eventually, the tools are combined to create something which involves problem solving. Abstraction comes from using the tools to create larger and more sophisticated programs. Creativity and expression come when you learn to use the small pieces and tools to read and write the world.

References

British Columbia Ministry of Education (2017). Applied Design, Skills and Technology Curriculum. Retrieved from https://curriculum.gov.bc.ca/curriculum/adst/K/core# on February 26th, 2018.

Buteau, C., Gadanidis, G., Lovric, M. & Muller, E. (2016). Report of the working group on Computational Thinking and Mathematics curriculum. In S. Oesterle, D. Allan, J. Holm (Eds.). *Proceedings of the 2016 CMESG/GCEDM Annual Meeting* (pp. 119-136). June, Kingston, ON. Retrieved from http://www.cmesg.org/wp-content/uploads/2017/07/CMESG-2016.pdf on February 21, 2018.

Clements, D. H. & Sarama, J. (2010). Learning Trajectories in Early Mathematics – Sequences of Acquisition and Teaching. In: Tremblay RE, Boivin M, Peters RDeV, eds. Bisanz J, topic ed. *Encyclopedia on Early Childhood Development*[online]. http://www.child-encyclopedia.com/numeracy/according-experts/learning-trajectories-early-mathematics-sequences-acquisition-and. Published July 2010. Accessed February 25, 2018.

Cuny-Snyder-Wing definition (2010). https://www.cs.cmu.edu/link/research-notebook-computational-thinking-what-and-why

Denning ,P. J. (2016). Remaining trouble spots with computational thinking. *Communications of the ACM*, *60*(6), 33-39.

Flake, G.W. (1998). *The computational beauty of nature*. MIT Press.

Gadanidis, G. (2017). Computer coding in the K-8 Mathematics curriculum. In What Works? Research into Practice. ON: Queen's Printer for Ontario.

Gadanidis, G. (2017). Five Affordances of Computational Thinking to support Elementary Mathematics Education. *Journal of Computers in Mathematics and Science Teaching, 36*(2), 143-151. Retrieved February 22, 2018 from https://www.learntechlib.org/p/174346/.

Global Math Challenge (n.d.). https://www.global-math.com

Harouni, H. (2015). Toward a political economy of mathematics education. *Harvard Educational Review, 85*(1), 50-74. https://doi.org/10.17763/haer.85.1.2q580625188983p6

Hewitt, D. (1999). Arbitrary and necessary part 1: A way of viewing the mathematics curriculum. *For the Learning of Mathematics, 19*(3), 2-9. http://www.jstor.org/stable/40248303

Jackiw, N. (2012). The geometer's sketchpad [Computer software]. Berkeley, CA: Key Curriculum Press.

Klarreich, E. (2015). Outsiders crack 50-year old math problem. *Quanta Magazine*, November, 24. Retreived from https://www.quantamagazine.org/computer-scientists-solve-kadison-singer-problem-20151124/ on February 24, 2018.

Kotsopoulos, D., Floyd, L., **Khan, S.**, Namakasa, I. K., Somanath, S., Weber, J. & Yiu, C. (2017). A pedagogical framework for computational thinking. *Digital Experiences in Mathematics Education*. DOI 10.1007/s40751-017-0031-2

Martinho-Truswell, A. (2018). To automate is human. Aeon, 13 February. Retrieved from https://aeon.co/essays/the-offloading-ape-the-human-is-the-beast-that-automates

Mueller, J., Beckett, D., Hennessey, E. & Shodiev, H. (2017). Assessing computational thinking across the curriculum. In P. J. Rich & C. B. Hodges (eds.). *Emerging Research, Practice and Policy on Computational Thinking* (pp. 251-267). Springer International Publishing. DOI: 10.1007/978-3-319-52691-1_16

Namukasa, I. K, Kotsopoulos, D, Floyd, L., Weber, J., Kafai, Y., Khan, S., Yiu, C., Morrison, L., Somanath, S. (2015). From computational thinking to computational participation: Towards Achieving Excellence through Coding in elementary schools. In *Proceedings of the Symposium on Math + Coding*. Western University, London, ON. http://www.researchideas.ca/coding/docs/CT-participation.pdf

Namukasa, I. K., Minakshi, P. & Miller, M. (2017). Tools for integrating computational thinking and mathematics in the middle grades. Math + Code 'Zine, 2(2). http://researchideas.ca/mc/ct-and-math-in-middle-grades/

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

Papert, S. (1996), An Exploration in the Space of Mathematics Education, *International Journal of Computers for Mathematical Learning*, 1(1), 95-123.

Pavlus, J. (2016). Searching for the algorithms underlying life. *Quanta Magazine*, January 28.Retrieved from https://www.quantamagazine.org/the-hidden-algorithms-underlying-life-20160128/ on February 24, 2018.

Polya, G. (1945). *How to solve it: A new aspect of mathematical method*. Princeton University Press.

Rozins, C. (2016). Problem solving through computational thinking. Presentation at Fields Institute Math Ed Forum.

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

Sinclair, N., & Patterson, M. (in press). The dynamic geometrization of computer programming. *Mathematical Thinking and Learning*.

Tatar, D., Harrison, S., Stewart, M. Frisina, C. & Musaeus, P. (2017). Proto-computational thinking: The uncomfortable underpinnings. In P. J. Rich & C. B. Hodges (Eds.) *Emerging Research, Practice, and Policy on Computational Thinking* (pp.63 - 84). Springer International Publishing

Tedre, M. & Denning, P. J. (2016). The long quest for Computational Thinking. In *Proceedings of the 16th Koli Calling Conference on Computing Education Research* (pp. 120-129), November 24-27, Koli, Finland. Retrieved from http://denninginstitute.com/pjd/PUBS/long-quest-ct.pdf on February 21, 2018.

Weintrop, D., Beheshti, E., Horn, M. et al. (2016). Defining computational thinking for mathematics and science classrooms. *Journal of Science Education and Technology, 25*(1), 127-147. https://doi.org/10.1007/s10956-015-9581-5

Wing, J. M. (2006), Computational Thinking, *Communications of the ACM*, Vol. 49(3):33–35, http://dx.doi.org/10.1145/1118178.1118215

Wing, J. M. (2011). Research Notebook: Computational Thinking-What and Why? The Link.