Number, arithmetic, multiplicative thinking and coding

Krista Francis, Brent Davis

Number, arithmetic, multiplicative thinking and coding

Krista Francis[1] and Brent Davis[2]

[1]University of Calgary, Werklund School of Education, Calgary, Canada; kfrancis@ucalgary.ca

[2]University of Calgary, Werklund School of Education, Calgary, Canada; brent.davis@ucalgary.ca

With frequent predictions of upcoming technological and economic difficulties triggered by an impending shortage of information and communications technologies (ICT) professionals, the calls are growing stronger to include coding as a core element of school curriculum. These calls are bolstered by the suggestion that coding supports the development of thinking skills – which echoes a longstanding argument for teaching mathematics. Motivated by the parallel, we attempted to investigate some of the common ground between learning to code and the development of core mathematical concepts. We photographed and video recorded children, aged 9–10, as they learned to build and program Lego Mindstorms™ EV3 robots over four days. Our findings suggest that programming supports children's understandings of decimal numbers and their transitions from additive to multiplicative thinking.

Keywords: Coding, robotics, arithmetic, number concepts, elementary education.

Introduction

In recent years there has been a growing recognition that information and communications technologies (ICT) are a major contributor to innovation and economic growth. For instance, the Organization for Economic Cooperation and Development (OECD, 2016) considers computer programming a necessity for a highly skilled labour force. Shortages are already felt across the world and demand for highly skilled ICT professionals is expected to rise. In our home country of Canada, for instance, there are predicted shortages of more than 150,000 skilled ICT workers in the next few years. This shortage is impacting IT innovations and revenues (see Arellano, 2015; Clendenin, 2014).

Canada is hardly unique on this count, as evidenced by major pushes around the world to include coding as a core part of school curriculum. In response, some educators and educational systems are shifting from teaching "how to use" software programs toward "how to code." Estonia and England, for example, have implemented a national curriculum that makes computer programming mandatory for all school-aged students across all grades, and other nations appear to be moving in this direction. For instance, it is currently a topic of political debate in Australia, where the opposition party is calling to have computer programming taught in every primary and secondary school in the country (Roumeliotis, 2015; Sterling, 2015).

In North America, national-level discussions and calls have yet to gather the same sort of momentum, but more and more initiatives are emerging at the local level. For example, the Chicago school district is adopting computer science as a core subject in all public high schools – prompted in large part by support from Google and Microsoft and through initiatives such as *Code.org* and *Hour of Code*, which are dedicated to expanding access to computer science for all U.S. students. Despite the absence of a national strategy in the U.S., messages on the importance of learning to code are frequent, with some emanating even from the President's office. In fact, coding skills have

been associated not only with empowering individuals and meeting employment needs, but with many aspects of the country's future and security (Pearce, 2013).

Trends toward including coding in school curriculum were preceded by a broadly effective worldwide push to get computers in schools. In 2011, most students (71%) in OECD countries reported having access to computers and the Internet at school. However, most students reported using the computers at school for email, browsing the Internet, word processing or doing individual homework. For the most part, such activities require low-level cognitive thinking and do not challenge students to develop more than basic user skills. Learning how to program a computer, it is typically argued, involves higher-level cognitive processes and provides opportunities for developing higher-level ICT skills.

These sorts of arguments for teaching computer coding parallel long-standing rationales for teaching mathematics. Similarly, many of the structures and strategies within coding bear strong resemblances to elements of mathematical concepts (Papert, 1980). We discuss a few of these resemblances in this paper, focusing on arithmetic.

Conceptual metaphors are one of the ways we understand mathematics (Núñez, 2000). With regard to the concept of number, Lakoff and Núñez (2000) describe "four fundamental metaphors of arithmetic": *arithmetic as object collection*, *arithmetic as object construction*, the *measuring stick metaphor*, and *arithmetic as object along a path*. The metaphor of *arithmetic as an object collection* is based on a one-one correspondence of numbers to physical objects. With this metaphor a greater size corresponds to a bigger number. For instance, 5 is *greater* than 2 because it forms a *bigger* collection. The metaphor of *arithmetic as object construction* is based on fitting objects/parts and arithmetic operations. For instance, 5 is greater than 2 because an object comprising 5 units is larger than one comprising two. The *measuring stick* metaphor maps numbers onto distances, whereby 5 is greater than 2 because it is *longer*. The metaphor of a*rithmetic as an object along the path* is based on arithmetic as motion, by which 5 is greater than 2 because it entails moving further from a common starting point (i.e., zero). Programing robots provides opportunities for illustrating and experiencing these arithmetic metaphors.

Context

In this interpretive study we asked what mathematics children learn by building and programming Lego Mindstorms^{TM} EV3. Interpretive research is about what meaning individuals construct in their lived experiences (Bhattacharya, 2008). We co-designed learning tasks with a graduate engineering student and co-taught the tasks with the classroom teachers over a course of four sequential days in three-hour daily sessions. The study's participants were 22 children, Grades 4-5 (aged 9–10), at Pakan School at Whitefish Lake 128 First Nation in rural Northern Alberta. Once the children knew the basic coding blocks for moving the robot, they were given Papert's (1980) task of programming a robot to follow a trace out of a triangle, square, pentagon or hexagon. On the third day, they were given the final challenge of building a robot that could find and douse a fire in any of four rooms in a building. Data included video-recordings, GoPro digital images, field notes, and artifacts including saved computer programs.

We video-recorded the four sessions to obtain rich contextual detail of children's mathematical interactions when programming the robots. Using interpretive video analysis (Knoblauch, 2013) we

selected videos and GoPro digital images that exemplified instances of children's mathematical thinking. Video data enables repeated viewing, slow motion, fast motion and frame-by-frame analysis. The selected videos formed the basis for emergent understandings of the children's experiences. The analysis developed through an iterative process of rereading the literature, reviewing the video and GoPro data, and rewriting. As is evident in our analysis, below, video data was vital. In particular, it permitted us to slow down the process and identify the integrated/nested processes of learning that occurred. The three instances that we use to focus our discussion were: (1) a trio of girls learning to program their robot for the final challenge to move a certain distance into the hallway to illustrate a developing understanding of number, (2) a boy tapping the vertices and sides of a triangle to count the number of programming steps necessary for the robot to move around the triangle as an example of additive thinking, and (3) a boy learning how the number of sides and angles of a polygon connects to the number of repeats in a loop, which illustrates a developing shift from additive to multiplicative thinking.

Findings

In the *numberline* video (see https://vimeo.com/144996708video), Krista was helping the pink team program their robot to move into the building. This action required manipulating one block of EV3 code to move the wheels a specified number of rotations. The team members started out with a guess of 0.4 rotations to move the robot into the first corridor of the building. After testing how far the robot moved and observing that the robot needed to move a considerably greater distance, Krista prompted the girls by asking what they should try next. Celina suggested they try 0.5. The small incremental change was still not enough, so Krista suggested they try 2. Two rotations moved the robot too far.

Krista: What is between 0.5 and 2?

Celina: 5.

Suspecting that Celina's response indicated that she and her teammates were unable to summon an appropriate interpretation of decimal numbers, Krista drew a simple number line on the whiteboard.

Krista: What is between 0.5 and 2?

Celina: Oh! 1.8.

The number Celina chose was close to the number of rotations actually required, which indicated she understood the meaning of 1.8. In the exchange above, we take Celina's immediate and satisfactory response to the repeated question as evidence that Krista was justified in her suspicion that the learners were lacking an appropriate interpretation for understanding decimal numbers – or, at least, were unable to extend whatever interpretations that had available to a situation in which distance was measured in wheel rotations. Coding the robot to move compelled the learners to elaborate their understandings. Invoking the number line appeared to provide an appropriate metaphor for helping Celina understand.

In the following sequence of images and descriptions, we summarize how the task of coding the robot to move into a room calls for all four of Lakoff and Núñez' (2000) representations of arithmetic. To begin, the metaphor of *arithmetic as an object collection* is used in most counting situations, whenever the forms being counting are perceived as discrete objects. It is by far the most

common interpretation of number through the task of assembling a robot, by simple virtue of the fact that the robots begin as large collections of separate items. Less obviously, it is also called for in coding moments as programmers translate complicated actions into discrete steps or instructions. And more obscurely too, such conceptual moves as the discretizing of wheel turns, so that they can be counted and thus used as a tool in programming, might be argued to rely on this metaphor. Figure 1 (left) presents an instance of this metaphor, showing that 2 turns is less (i.e., forms a smaller set than) 5 turns.

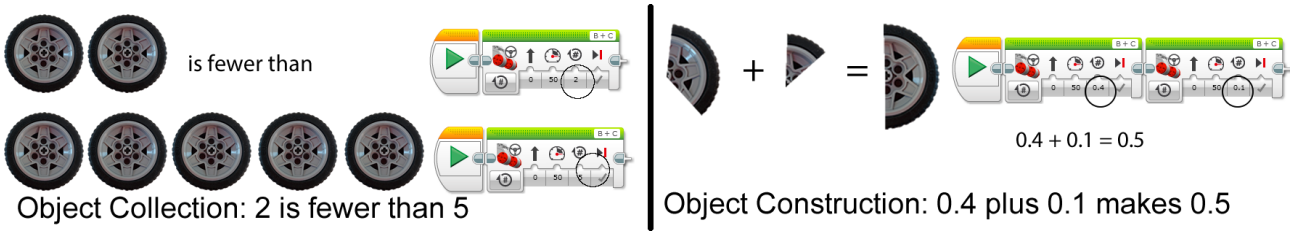

Figure 1: Arithmetic as object collection. Number of wheel rotations | Arithmetic as Object construction – combining portions of wheel rotations into single objects

Figure 1 (right) shows how the metaphor of *arithmetic as an object construction* might be encountered when programming a robot to move. Celina wanted a larger wheel rotation than 0.4, so she added an incremental amount of 0.1 wheel rotations to make 0.5 wheel rotations. Contrasted to the previous metaphor, in this instance, wheel turns are not perceived as discrete objects, but as parseable continuities. Those parsed elements can then be assembled into an appropriate object to move the robot a precise distance.

The *measuring stick* metaphor also featured prominently in the children's programming, and was particularly prominent in the frequent need to interpret wheel turns in terms of actual distances (e.g., when the phrase "1 wheel turn" was deployed not as a description of movement but was a reference to a distance of roughly 12 cm). Figure 2 (left) in reference to the instance in which the room of the hall was shorter than approximately 1.8 wheel turns. In this instance, programming the code block requires understanding measurement.

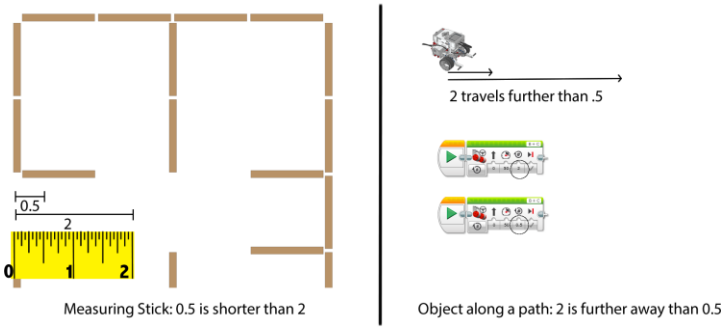

Figure 2: Measuring Stick: The length of hall | Arithmetic as an object along the path. The robot travels further with 2 than 0.5

Figure 2 (right) shows how programing the robot to move draws upon the metaphor of *arithmetic as on object along the path.* In this case, starting place becomes a critical element is that, for example, occurs when the robot enters the room and recurs in the opposite direction when the robot leaves.

To re-emphasize, we observed each of Lakoff and Núñez' four metaphors of arithmetic to be present in programming the robot to move a required distance in the room. The ability to identify to the particular metaphor(s) that a situation is calling for is a critically important teaching competence, as Krista demonstrated in the interaction with Celina. Re-interpreting that brief episode, Krista recognized that Celina was not interpreting number as a distance (i.e., she was not using a *measuring stick* metaphor), and thus reminded her of that metaphor by offering the image of a number line. No explanation other than an image of number that fitted the application at hand was required.

Arithmetic Topic 2 – Moving from "additive" thinking to "multiplicative" thinking.

The need for appropriate metaphors and images of number isn't sufficient for making sense of that entire episode, however, closer analysis reveals a further issue with the children's arithmetic, namely the tendency to default to additive actions rather when multiplicative actions would have been more suitable. That episode began with the group's realization that an entry of "0.4" moved the robot only a small portion of a desired distance. Asked what else they might try, they increased the distance only incrementally by 0.1 (to 0.5) rather than the necessary factor of (roughly) three.

This same tendency to default to additive actions when multiplicative action would have been more productive was witnessed many times across many groups over the four-day project. The *additive thinking* video (see https://vimeo.com/144820583) provides a window into any instance of the same phenomenon. In this case, Gene, who was on the floor in orange, is figuring out how many blocks of code were needed for the program. As he counted "one, two, three, four, five, six," he tapped each vertex and side of the yellow triangle, finally announcing that six steps are needed. Gene's step-by-step of the same two steps (straight, turn, straight, turn, straight, turn sequence) is an example of additive thinking – that is, of construing the situation in terms of a sequence of increments rather than a repetition.

Phrased in terms of coding, Gene opted to repeat the same line of instructions six times rather than employing a loop that ran six times. This happened in spite the fact that he and his group mates had learned how to use loops the day before when they programmed their robot to dance.

In fact, only one of the 8 groups in the class used a loop for the polygon task – suggesting that the move from additive/increment-based thinking to multiplicative/loop-based thinking is more conceptually demanding than is often assumed. The *additive to multiplicative thinking* video (see https://vimeo.com/144826969) further illustrates this point, as the classroom teacher along with Krista attempted to help Liam program with loops. Liam, on the left, identified that a pentagon has three sides. When asked to count the sides, he walked around the pentagon counting aloud and announced "5 times." Krista explains that 5 times is the number of times to repeat the two block codes (go straight and turn) in the loop. In response, Liam exclaimed excitedly, "Yes!"

In the same clip there are two boys who were fine-tuning their robot's program to follow a triangle. Their robot never stopped, which indicates that they are using an infinite loop – suggesting that they are making use of a concept of "repeating," but likely not a concept of multiplication. After three attempts at tracing out a triangle, they still hadn't crafted a program that would stop their robots.

Davis and Renert (2014) have identified a number of common instantiations for multiplication that are encountered in elementary school classrooms, including grouping, hopping, repeated sums,

stretching and compressing, array- and area-making, and making combinations. Looping, it seems, is another, distinct instantiation of multiplication that is particularly powerful in the activity of programming – in a manner, we suspect, that might be used reflexively to support mathematics learning. Figure 3 below, illustrates two programs for following a triangle. Additive thinking is found with the sequential accumulation of six programming blocks: move forward, turn, move forward, turn, move forward turn. Multiplicative thinking requires recognizing that the triangle can be traced by repeating the move forward and turn blocks three times in a loop. In the exchanges above, Liam appeared to be developing fluency with multiplicative thinking.

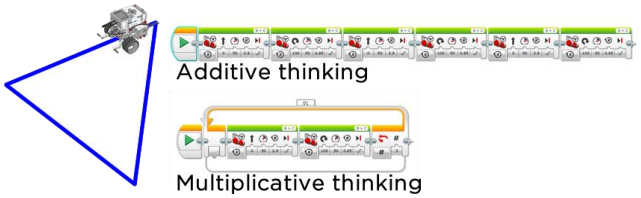

Figure 3: An additive and a multiplicative program for moving a robot in a triangle

Across the participants there was a pervasive tendency to program robots to trace out polygons as a sequence of same-steps rather than as a repetition of a set of steps (i.e., as enabled with a loop). This tendency was not easily interrupted through instruction, which provides evidence of the complexity of thinking multiplicatively. Even at the end of the four days, during the final challenge, only two of the teams had managed to appreciate the power of loops sufficiently to incorporate them into their programs. Not surprisingly, theirs were also the robots that performed the best. In one of these cases, the code for the winning robot (see https://vimeo.com/145404678) involved a loop determining if a fire is present, announcing "Yes" or "No" as appropriate, and activating an arm motion to dump retardant if "Yes."

Part of the reason that we dwell on this point is that the operation of multiplication is, arguably, the most important concept in grade-school mathematics. Multiplicative thinking is the cornerstone of proportional thinking, which is foundational to advanced mathematics for reasons that include the access it affords to an extended range of numbers (for example, larger whole numbers, decimals, common fractions, ratio and percent), its role in recognizing and solving a range of problems involving direct and indirect proportions, and the power it offers with its prominent place in school-based concepts and processes (Education and Training, 2013). In brief, multiplicative thinking is a key in the transition from early ideas to later ideas (see, e.g., ACME, 2011, p. 20).

Closing remarks

Our preliminary findings suggest programming robots can support learning mathematics. In the episodes reported, the tasks of programming robots required more than parsing complicated actions into singular direction; they entailed flexible engagement, Lakoff and Núñez' (2000) conceptual metaphors and mathematical models.

Computer programming aligns closely with concepts and structures in mathematics and we suspect that it might provide other powerful instantiations for mathematical concepts that have not yet been noticed. That suggestion is perhaps not surprising, given the mathematical roots of computer programming. However, to our reading, it is not an aspect of programming that has garnered much consideration in either mathematics education or the technology education literature. Considering

that mathematics literacy and competency with coding are of growing relevance, engagement with emergent technologies can complement and co-amplify mathematics learning, and contribute to evolving understandings of what "basic" mathematics might be for our era.

With regard to important complementarities between learning mathematics and learning to code, the Lego Mindstorms™ EV3 robots and the associated programming language provide a powerful instance of multiple solutions. They afford tremendous flexibility for accomplishing a range of tasks, from the trivial to the complex. None of the coding tasks set for the children in our study had pregiven or optimal "solutions." Despite that – or perhaps because of that – the children were able to engage in manners that they could recognize as successful, even when "complete" solutions were not reached. With incremental tasks and iterative refinements, children were able to learn more sophisticated and efficient methods for programming the robot. It is not difficult to imagine a mathematics class with similar standards of success.

That said, it is not a coincidence that the winning robot had the most efficient and sophisticated program of the group. Some answers are better than others, and those answers appear to reflect powerful mathematical thinking. Our future longitudinal research will investigate how children's understandings of mathematical concepts and programming robotics develop over several years.

We believe that the results of this study underscore the importance of developing and implementing a computer programming curriculum in schools. Coding is an emergent literacy that can amplify other critical literacies, while affording access to a diverse range of cultural capitals. The reasons to teach coding go beyond the technical and economic; for us, they are fundamentally ethical.

References

ACME. (2011). *Mathematical needs: The mathematical* needs *of learners* (p. 27). London, UK: The Advisory Committee on Mathematics Education. Retrieved from http://www.acme-uk.org/media/7627/acme_theme_b_final.pdf

Arellano, N. E. (2015). Why Canada has an 182,000 IT talent shortage while lots of tech professionals are out of work. *IT World Canada*. Retrieved from http://www.itworldcanada.com/article/why-canada-has-an-182000-it-talent-shortage-while-lots-of-tech-professionals-are-out-of-work/373517

Bhattacharya, H. (2008). Interpretive research. In L. Given, *The SAGE encyclopedia of qualitative research methods*. Thousand Oaks, CA: SAGE Publications.

Clendenin, B. (2014). Canada's IT labour shortage: Challenges and opportunities. *IT Business*. Retrieved from http://www.itbusiness.ca/blog/canadas-it-labour-shortage-challenges-and-opportunities/50250

Davis, B., & Renert, M. (2014). *The math teachers know: Profound understanding of emergent mathematics.* New York, NY: Routledge.

Education and Training. (2013). *Multiplicative thinking*. Melbourne, Australia: Victoria State Gorvernment. Retrieved from http://www.education.vic.gov.au/school/teachers/teachingresources/discipline/maths/assessment/pages/multithink.aspx

Knoblauch, H. (2013). Videography. Focused ethnography and video analysis. In H. Knoblauch, B. Schettker, & J. Raab (Eds.), *Video analysis : Methodology and methods: qualitative audiovisual data analysis in sociology* (3rd ed., pp. 69–84). Frankfurt: Peter Lang.

Núñez, R. E. (2000). Mathematical Idea Analysis: What Embodied Cognitive Science Can Say about the Human Nature of Mathematics. In *Proceedings of the Conference of the International Group for the Psychology of Mathematics Education* (Vol. 1, pp. 3–22).

Lakoff, G., & Núñez, R. E. (2000). *Where mathematics comes from: how the embodied mind brings mathematics into being*. New York, NY: Basic Books.

OECD. (2016). *Skills for the Digital World*. Paris, France: Organisation for Economic Co-operation and Development. Retrieved from http://www.oecd.org/officialdocuments/publicdisplaydocumentpdf/?cote=DSTI/ICCP/IIS(2015)10/FINAL&docLanguage=En

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

Pearce, K. (2013). Why You Should Learn To Code (And How To Do It!). Retrieved from http://www.diygenius.com/learn-to-code-online/

Roumeliotis, I. (2015). Back to school: Canada lagging in push to teach kids computer coding. *CBC New*. Canada. Retrieved from http://www.cbc.ca/news/technology/back-to-school-canada-lagging-in-push-to-teach-kids-computer-coding-1.3185926

Sterling, L. (2015). An education for the 21st century means teaching coding in schools. *The Conversation*. Retrieved from http://theconversation.com/an-education-for-the-21st-century-means-teaching-coding-in-schools-42046

Acknowledgments

We thank participating teachers and students. We are also grateful to Michael Poscente for designing and coaching the Lego™ robotics task. This work was funded by the Imperial Oil Science Technology Engineering and Mathematics Education Initiative.