ALGORITHMIC THINKING: AN INITIAL CHARACTERIZATION OF COMPUTATIONAL THINKING IN MATHEMATICS

Elise Lockwood
Oregon State University
Elise.Lockwood@oregonstate.edu

Autumn Asay
Oregon State University
asayb@oregonstate.edu

Anna F. DeJarnette
University of Cincinnati
dejarnaa@ucmail.uc.edu

Matthew Thomas
Ithaca College
mthomas7@ithaca.edu

Computer scientists have reported on "computational thinking," which Aho (2012) defines as "the thought process involved in formulating problems so their solutions can be represented as computational steps and algorithms" (p. 832). We wanted to investigate such thinking in mathematics. To study this, we interviewed five mathematicians about the role of computation in their work, and the notion of "algorithmic thinking" developed from these interviews. We found that mathematicians value this thinking, and in this paper we define algorithmic thinking and present a number of applications and affordances of such thinking.

Keywords: Post-Secondary Education, Cognition, Reasoning and Proof

Introduction and Motivation

Suppose we asked you how many ways there are to arrange the letters in the word COMPUTER, and, even more, we asked you to sketch out an algorithm that would enumerate an organized list of all the arrangements. Or, considering other contexts, suppose we asked you to write a program to compute the standard deviation of a data set or to shift a graph of the function $y = sin(x)$ to the right by π radians. Is there a certain way of thinking, a type of knowledge, or a set of skills that you would use to successfully complete such tasks?

These kinds of questions motivate our work, and broadly we seek to explore the ways that technological tools shape the way that people think about and do mathematics. One component of this is to consider the role of procedures and algorithms in doing mathematics, as well as the kinds of thinking that might underpin the design and implementation of such procedures. In the literature, there are ongoing conversations about the nature of procedural knowledge (e.g., Hiebert & Lefevre, 1981; Baroody, et al., 2007; Star 2005, 2007), and there have been explicit calls to foreground the study of procedural knowledge so that it may be better understood (Star, 2007). As a preliminary step toward understanding such thinking, we interviewed mathematicians about the role of computation and algorithms in their work, whether they think there are certain ways of thinking or activity to help students learn and successfully implement computation, and whether or not such thinking is important. By interviewing mathematicians, our goal is to provide evidence that members of the mathematical community value such thinking and that it may be beneficial to foster such thinking among students. We consider these findings a proof of concept, supporting the idea that such thinking is of value to members of the mathematical community and is worthy of further investigation. Specifically, our research questions are: a) How do mathematicians characterize the kind of thinking that facilitates computation in their work, and b) what are applications and affordances of such thinking?

Background Literature

In computer science, there is a construct called *computational thinking* (CT) (Grover & Pea, 2013; Wing, 2006; 2008), which Wing initially described as "taking an approach to solving problems, designing systems and understanding human behaviour that draws on concepts

Wood, M. B., Turner, E. E., Civil, M., & Eli, J. A. (Eds.). (2016). *Proceedings of the 38th annual meeting of the North American Chapter of the International Group for the Psychology of Mathematics Education*. Tucson, AZ: The University of Arizona.

fundamental to computer science" (2006, p. 33). Wing went on to characterize CT broadly and as encompassing many kinds of thinking and activity, such as "thinking recursively" (p. 33), "using abstraction and decomposition when attacking a large complex task or designing a large complex system" (p. 33), "using heuristic reasoning to discover a solution" (p. 34), and "making trade-offs between time and space and between processing power and storage capacity" (p. 34). This broad characterization makes it difficult to pin down a precise definition, but it provides a start for identifying common threads among computational activity, suggesting that perhaps there are common ways of thinking that underlie reasoning about computation and algorithms. Aho (2012) built on these definitions, and he considered "computational thinking to be the thought process involved in formulating problems so their solutions can be represented as computational steps and algorithms" (p. 832). These computer scientists seem to be articulating a way of thinking that they view as essential to the kind of work they do in their field, and these initial definitions of computational thinking as inspiring this study. There has also been some work in STEM education related to computational thinking (e.g., Weintrop, Beheshti, Horn, Orton, Trouille, Jona, & Wilensky, 2014; Wilensky & Reisman, 2006). We are motivated to understand whether there is some version of computational thinking that might be applied in a mathematical context.

Theoretical Framework

Conceptual and Procedural Knowledge

Hiebert and Lefevre (1981) elaborate an important distinction between conceptual knowledge and procedural knowledge, and since then many more mathematics education researchers have contributed to the discussion on the topic (e.g., Baroody, Feil, & Johnson, 2007; Star, 2005, 2007). Hiebert and Lefevre characterize *conceptual knowledge* as "knowledge that is rich in relationships. It can be thought of as a connected web of knowledge, a network in which the linking relationships are as prominent as the discrete pieces of information" (p. 3-4). They define *procedural knowledge* as being made up of two parts: "One part is composed of the formal language, or symbolic representation, of mathematics. The other part consists of the algorithms, or rules, for completing mathematical tasks" (p. 6). Hiebert and Lefevre (1986) also made a distinction between meaningful and rote learning. They claim that "meaning is generated as relationships between units of knowledge are organized or created," (p. 8) and noted that by their definition, conceptual knowledge must be learned meaningfully (p. 8). Rote learning "produces knowledge that is notably absent in relationships and is tied closely to the context in which it is learned" (p. 8), and they state that conceptual knowledge cannot be developed by rote learning, while procedural knowledge can. Although procedural knowledge and rote knowledge ought not to be conflated, there has been a tendency to characterize procedural knowledge as being shallow. As Star (2005) points out, Hiebert and Lefevre characterize the sequential nature of relationships in procedural knowledge, and "by this definition, procedural knowledge is superficial; it is not rich in connections" (p. 407). Star (2005, 2007) argues that this characterization, in part, has contributed to perhaps a devaluing of procedural knowledge and resulted in a lack of empirical research on procedural knowledge.

We acknowledge that there is some debate as to the nature of procedural knowledge, and that procedure knowledge has typically been related to students' implementation of known procedures. The mathematicians we interviewed talked about not just the implementation of procedures, but also the development and designing of procedures and algorithms. We are interested in studying the kind of thinking and knowledge involved in designing procedures, and we wonder whether such knowledge is related to (or is an extension of) procedural knowledge as it is currently characterized. With these ideas in mind, the constructs of conceptual and procedural knowledge helped to frame our study. We hope that our findings can contribute to Star's narrative that procedural knowledge can be deep and important, ultimately providing insight about why and how such thinking might develop.

Wood, M. B., Turner, E. E., Civil, M., & Eli, J. A. (Eds.). (2016). *Proceedings of the 38th annual meeting of the North American Chapter of the International Group for the Psychology of Mathematics Education*. Tucson, AZ: The University of Arizona.

Methods

To accomplish these goals, we interviewed five mathematicians in single 60-90 minute semi-structured interviews. Four of the mathematicians were professors in mathematics departments (M1 was a mathematical biologist, M2 an applied mathematician who specialized in modeling, M3 a numerical analyst, and M4 a geometer), all holding PhDs in mathematics, and one worked in industry and had a Master's degree in statistics. All interviews were audio-recorded. We were motivated by the literature on computational thinking (Wing, 2006, 2008), and we initially framed the interviews in terms of better understanding the role of computation. Thus, we asked the mathematicians to reflect upon various aspects of computation, including computation in their own work, the value of computation for themselves and for students, how they might teach computation, and whether there were particular kinds of thought that might support computation. Because terms like "computational thinking" or "algorithmic thinking" were not likely to carry much meaning with the mathematicians initially, and because of the various areas of mathematical expertise, we sought to establish some common language for each interviewee.

The first half of the interview was thus spent asking mathematicians about their own particular work, and especially for examples of computation in their own work. The interviewer could then follow up with appropriate questions that could target particular questions about (what we now term as) algorithmic thinking. For example, after asking some preliminary demographic questions, we asked the following: *Do you use computation in the work that you do? How so? How are you defining 'computation'?* and *What are some specific ways (or contexts) in which you use computation in your work?* This enabled us to get a sense of how they viewed and might use computation. When we had an example, we could ask questions such as *Is there a certain type of thinking/behavior/activity that needs to be carried out when performing such a mathematical computation?* This was the crux of what we wanted to study in the interview, and we could adapt this question based on specific examples a certain mathematician would give. Once this kind of language and discussion was established, we could ask them for preliminary definitions of "computational thinking" and "algorithmic thinking." We also asked whether they thought computation is important for mathematicians to learn, and why or why not. We concluded with discussions about whether and how they had taught computation before, and for them to weigh in on how students might learn computation.

To analyze the data, we transcribed the interviews, and members of the research team organized responses to the various questions. Then, using a constant comparative method (Strauss & Corbin, 1998), we generated categories of relevant phenomena that emerged from passes through the transcripts, resulting in themes that shaped a narrative to describe the results.

Results

We remind the reader that our main research question was to investigate how mathematicians characterize the kind of thinking that facilitates computation in their work, and we were especially interested in studying whether or not *computational thinking* might be a meaningful term for mathematicians. Throughout the interviews, though, it became clear that the mathematicians found computational thinking to be too broad to characterize meaningfully, but that the term *algorithmic thinking* provided more appropriate language to describe the kind of thinking we targeted in the interviews. It is important to emphasize that the term algorithmic thinking is something that emerged from, and seemed to resonate with, the interviewees. In this results section, we first define and characterize algorithmic thinking and describe how it emerged from the interviews with the mathematicians. Then, we outline two reasons why mathematicians seem to value such thinking: first, because it has a variety of practical applications that they use in their work, and second, because such thinking offers broader implicit affordances that are desirable for doing mathematics. With

Wood, M. B., Turner, E. E., Civil, M., & Eli, J. A. (Eds.). (2016). *Proceedings of the 38th annual meeting of the North American Chapter of the International Group for the Psychology of Mathematics Education*. Tucson, AZ: The University of Arizona.

these findings, we hope to convince the reader that algorithmic thinking is a specific, useful construct that is worth trying to engender in students.

Algorithmic Thinking

As noted in the methods, in each interview there was some effort required in getting on the same page with the mathematicians, having them articulate the kinds of computation they do in their work and the kind of knowledge, skills, or training might be necessary for such work. These conversations gave rise to the following preliminary, working definition of algorithmic thinking: *A logical, organized way of thinking used to break down a complicated goal into a series of (ordered) steps using available tools*. We note that our definition has similarities with Aho's (2012) characterization of computational thinking, although we emphasize using procedural steps to solve a given goal, and he emphasizes formulating problems so they may be solved with computational steps. To see how this definition emerged from the data and to elaborate various aspects of this definition, we highlight some of the mathematicians' responses. Due to space constraints, we report on M1's response in detail and only briefly mention comments from some of the other mathematicians.

The term "algorithmic thinking" first emerged in our interview with M1, whose work involves analyzing biological data. In describing how computation arises in his work, he noted that after familiarizing himself with data, he will ask himself a question like "Okay, what sort of math tools or what sort of algorithmic approaches would be fit for this approach?" Then, after implementation on test data, he might engage in an iterative process, asking himself "Okay, how can we modify this algorithm to make it work a little bit better?" M1's work involved not just computation, but in particular the development, implementing, and refining of algorithms.

We eventually asked M1 if he could define computational thinking or algorithmic thinking. He said, "computational thinking makes me think of somebody that has an eye to computation" and went on to say that, "algorithmic thinking is something I think that I could try to describe a little bit more." He gave the following definition of algorithmic thinking: "Yeah, so, I have only just today verbally articulated, but really like, the idea of how to practically solve a goal by breaking it down from a series of steps that consist only of the tools that you have at hand."

To clarify what he meant by "only the tools you have at hand," he noted that there are some restrictions depending on the context or environment in which one is creating an algorithm. In one context, like Matlab, there may be powerful packages that one can evoke, whereas in Mathematica the commands might need to be more specific. Thus, the algorithm must be created given whatever restricted set of tools is available. We have incorporated some of his ideas into our definition above, highlighting the notion of a "break down" and the idea "available tools" from M1's definition. Prior to our interview with M1, the term "algorithmic thinking" had not been something we were necessarily targeting (rather, we were interested in "computational thinking"). Through his interview, however, we realized that this might be an appropriate and meaningful term for the mathematicians. Because a working definition of "algorithmic thinking" emerged from the first interview, we subsequently asked about it in the remaining interviews.

M3 highlighted two features of algorithmic thinking – compartmentalization and order. He talked about compartmentalization as being able to "see the big picture but also recognize the smaller blocks." He went on to say, "It's the same thing when you're looking at code. It might be written linearly, but each chunk does different things." He also emphasized the importance of putting things in the proper order. He said, "The big picture has maybe three blocks. Those obviously have to be in the proper order, but the farther you zoom in, you still need to make sure each subsequent subsection is in the proper order." M5 also articulated the series of steps and the compartmentalization that M3 alluded to.

Wood, M. B., Turner, E. E., Civil, M., & Eli, J. A. (Eds.). (2016). *Proceedings of the 38th annual meeting of the North American Chapter of the International Group for the Psychology of Mathematics Education*. Tucson, AZ: The University of Arizona.

M5: I guess the algorithmic thinking is something I can relate to, based on the conversations we just had. Is that you have an idea of the series of steps that you need to go through to analyze a data set. And each one of those steps could then be decomposed into more steps.

The language of "ordered steps" in our statement is meant to convey the structured process that M3 and M5 articulate, and M3's compartmentalization echoes M2's "break down" language.

In our interview with M4, we were discussing the activity of writing code (specifically designing, as opposed to simply implementing, a package in Geometer's Sketchpad). We asked him, "What does writing the package bring that is beneficial?" and he responded, "So one of the things that it brings is these general critical thinking for programming skills." We then pressed him on what "critical thinking for programming" might mean, and he said, "Um, it's a very linear structure, a linear representation that I'll let, for the lack of a better name I'll call coding, um, that requires a particular form of creative, logical reasoning." Later, when discussing what the term "algorithmic thinking" might mean, M4 noted that such thinking was closely tied in to his notation of critical thinking for programming. We interpret then that his sense of algorithmic thinking involves a linear structure that requires logical reasoning.

The mathematicians also indicated that algorithmic thinking transcended particular programming environments or particular contexts. For example, M1 said, "Yeah, I want them to gain the skills, and I don't really care what the language is. Because once—it's really surprising. So once you've steeped yourself in one language and really understand the process of algorithm design and approximating mathematical quantities and practically implementing it, those same skill sets transfer over real quickly to some other language." M4 had a similar sentiment about the overarching nature of algorithmic thinking: "What I care about is that they develop both an appreciation for, and an ability with, interacting with a machine to problem-solve. And that they learn how to write some code, not just ask one-line questions, so that they have an experience with debugging code. Once they've done that for any language, they can build on that in any context they need it later." We feel that these comments are important because they highlight algorithmic thinking as a way of thinking that may be used and applied in a variety of contexts. They also underscore the fact that there is some kind of broader way of thinking that may be associated with engagement with designing and implementing procedures and algorithms.

To summarize, the mathematicians articulated some key features of the kind of thinking that might be involved in computation that entails writing, implementing, and checking algorithms. We address further issues related to the nature of algorithmic thinking in the Discussion section.

Practical Applications of Algorithmic Thinking

In the interviews, the mathematicians articulated several of the ways that they use computation and algorithms in their work. In this section, we use the term computation to mean the kind of computation that occurs in conjunction with algorithmic thinking (i.e., computation involving the design and implementation of procedures and algorithms). Without exception, the mathematicians saw algorithmic thinking as being useful. The following comments by M1 are representative of the opinions of a number of the mathematicians, which are that the ability to design and implement algorithms is a vital part of being a mathematician:

M1: Honestly with how data-driven [it] is and [is] increasingly becoming, and how everything Is electronics- and computer-based. I think it's pretty vital to have that, even if you're going to be some totally pure mathematician or field—like number theory or something like that—eventually there will be a real-world application. … Even being aware and having a little bit of exposure and practice of that sort of theory to practice algorithmic implementation and computationally thinking is, I think, critical and/or it would be an incomplete education without having it.

Wood, M. B., Turner, E. E., Civil, M., & Eli, J. A. (Eds.). (2016). *Proceedings of the 38th annual meeting of the North American Chapter of the International Group for the Psychology of Mathematics Education*. Tucson, AZ: The University of Arizona.

On the whole, the mathematicians articulated that such computation is useful because it allows them to accomplish some things that cannot be done efficiently by hand. Here we briefly mention examples of practical applications of thinking that emerged in the interviews.

First, some of the mathematicians seemed to frame such computation as a tool allowing them to do mathematics, which is perhaps not surprising. Here, computation is framed as a means to an end, something that facilitates the exploration of a conjecture or the generation of examples that ultimately serve the purpose of informing a theoretical mathematical exploration. M4 discussed his use of Mathematica, and he said, "the more Mathematica I have learned, the easier it has been to perform the complicated computations. But all of that elegance in programming is just to make my life easier. There's nothing inherently, um, there's, there's no direct contribution to the research project or the results that comes from my ability to program in Mathematica, beyond the fact that there is a way for me to check these difficult computations." His comment that the "elegance in programming is just to make my life easier" suggests a perspective of programming as a tool to streamline the mathematical exploration he needs to conduct. He then made the following comment, which again highlights the relationship between the role of computation and how it relates to his mathematical work.

> *M4*: So it's an interaction. I'm, I'm using Mathematica as a crutch to do my computations for me as I am thinking through which calculations to do…So I try things and see what happens. And Mathematica gives me a way to tell what's happening. And sometimes it's productive and sometimes it isn't. Um, there's a little bit of each. There are times when I know the answer and I'm just checking that it's true.

M1 and M5 offered a different kind of practical application, one in which the algorithm is itself an integral part of the work of analyzing data. M1 described a practical application of needing to refine a particular algorithm for the purpose of analyzing biological data: "Yeah, yeah, so basically there's this math quantity that I want to compute, called the earth-mover's distance, and there exists algorithms to compute it exactly, but they're too slow. So I'm going to approximate that actual metric in a heuristic, algorithmic fashion." Here the act of developing an algorithm is not just for facilitating mathematical exploration, but rather it itself is a fundamental aspect of the mathematical activity. M5 similarly used algorithms in order to analyze data statistically: "Well, yeah, I do use computation in that I analyze data for a living. And most of that is through a series of algorithms that I'll run on a set of data that can range from doing simple statistics to executing predictive models to graphics." We consider these examples as practical ways that algorithmic thinking can be used in both pure and applied mathematics, highlighting that mathematicians do draw upon such thinking that allows for a variety of applications across a number of contexts.

Implicit Affordances of Algorithmic Thinking

In addition to the practical applications for which algorithmic thinking might be useful, the mathematicians also suggested a number of other potential benefits of algorithmic thinking. This emerged through explicitly asking the mathematicians about whether and why students should develop this kind of thinking and why such computation is important for students to learn. There are three major affordances that emerged from the interviews.

First, algorithmic thinking seems related to mathematical practices such as proving and problem solving. For example, M1 said the following of algorithmic thinking: "It's problem solving. And it's, at its core, the same problem solving that mathematicians do all the time, which is "Here are the constraints, my assumptions. This is the goal, my theorem. Now what steps can I break it down to get over there?" And mathematicians do this all the time. It's just that the set of tools that they have are totally different than when you want to apply that same process to a computational problem." We infer that M1 is suggesting that the kind of thinking involved in designing and implementing an

Wood, M. B., Turner, E. E., Civil, M., & Eli, J. A. (Eds.). (2016). *Proceedings of the 38th annual meeting of the North American Chapter of the International Group for the Psychology of Mathematics Education*. Tucson, AZ: The University of Arizona.

algorithm is closely aligned with mathematical problem solving, and thus that reinforcing algorithmic thinking could help to strengthen problem solving.

Second, M3 made interesting comments about the relationship between algorithmic thinking and communicating mathematical ideas with others. He said, "you couldn't make clear arguments to convince people of other types of conclusions that you're trying to make if you didn't take your arguments through logical steps. So if you're trying to convince anybody of something, you need to tell them that your solution, or your idea, does what you think it does and nothing else. And that's exactly what a code is supposed to do, too." He went on to say, "I'm saying that it, it not only helps with doing math. It also helps with communicating math." For his students, then, he saw that beyond whatever mathematical insight they were gaining, such thinking could help improve their mathematical communication skills.

Finally, a number of the mathematicians also talked about the importance of having a certain disposition toward being wrong and fixing mistakes, and they suggested a relationship between this idea and algorithmic thinking. For example, M2 said, "Well, of course, and there is the other ability of being able to recover from mistakes. Which, in computing, is fundamental. And not being too frustrated and just keep going back and forth. And trying to morph something that you know worked to something you know should work." Other mathematicians also suggested that debugging code is a key activity in working with algorithms, and we contend that debugging may help to engender the ability to recover from mistakes that M2 thinks is important.

In sum, the mathematicians articulated a number of affordances of algorithmic thinking that extend beyond practical applications. These results provide additional reasons for why mathematicians value such thinking and believe it is important for students to develop.

Discussion

An important note about our definition of algorithmic thinking is that in some ways it is similar to the kind of procedural knowledge that Hiebert and Lefevre define, as they highlight procedural knowledge as involving a sequential nature of steps: "A key feature of procedures is that they are executed in a predetermined linear sequence. It is the clearly sequential nature of procedures that probably sets them most apart from other forms of knowledge" (p. 6). However, our definition of algorithmic thinking goes beyond just the implementation of a procedure, or even the explanation of why a procedure works as it does. It involves planning and designing the steps, and having an overall sense of what the algorithm can do, as well as having the details in place to successfully implement the algorithm.

In this way, the mathematicians' characterization of algorithmic thinking may be related to Star's (2005) definition of deep procedural knowledge, which involves "knowledge of procedures that is associated with comprehension, flexibility, and critical judgment" (p. 408). In particular, algorithmic thinking seems to involve a decision-making process about what tools to use and how to organize them, and this necessarily involves a flexible way of thinking. We suppose, then, that this notion of algorithmic thinking may be conceived of as overlapping with procedural knowledge. Indeed, we argue that an understanding of how to design (and not just implement) an algorithm may be a hallmark of procedural knowledge that could be considered to be more than just superficial. We also see our work as contributing because we highlight what might be entailed in algorithmic thinking at a more advanced level and as a disciplinary mathematical practice. This offers another perspective on procedural knowledge, allowing another perspective to flesh out nuances of this important construct.

Conclusion and Avenues for Future Research

Although this work is preliminary, we believe that the mathematicians' responses suggest that it may be promising to pursue and refine this idea of algorithmic thinking and what it might entail, as mathematicians seem to believe that algorithmic thinking is a valuable component of the practice of

Wood, M. B., Turner, E. E., Civil, M., & Eli, J. A. (Eds.). (2016). *Proceedings of the 38th annual meeting of the North American Chapter of the International Group for the Psychology of Mathematics Education*. Tucson, AZ: The University of Arizona.

doing mathematics. In light of this, it is important for researchers to think through how we might help students develop this way of thinking. In terms of next steps, we hope to continue to investigate the nature of algorithmic thinking. In particular, we want to gather more evidence about algorithmic thinking from both mathematicians and students at a variety of levels, with the ultimate goal of better understanding the relationship between (deep) procedural knowledge and algorithmic thinking as well as how to engender such thinking among students.

References

Aho, A. V. (2012). Computation and computational thinking. *The Computer Journal, 55*(7), 832-835.

Baroody, A. J., Feil, Y., & Johnson, A. R. (2007). An alternative reconceptualization of procedural and conceptual knowledge. *Journal for Research in Mathematics Education, 38*(2), 115-131.

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

Hiebert, J., & Lefevre, P. (1986). Conceptual and procedural knowledge in mathematics: An introductory analysis. In J. Hiebert (Ed.), *Conceptual and procedural knowledge: The case of mathematics* (pp. 1-27). Hillsdale, NJ: Lawrence Erlbaum.

Star, J. R. (2005). Reconceptualizing procedural knowledge. *Journal for Research in Mathematics Education, 36* (5), 404-411.

Star, J. R. (2008). Foregrounding procedural knowledge. *Journal for Research in Mathematics Education, 38*(2), 132-135.

Weintrop, D., Beheshti, E., Horn, M. S., Orton, K., Trouille, L., Jona, K, & Wilensky, U. (2014). Interactive assessment tools for computational thinking in high school STEM classrooms. In D. Reidsma et al., (Eds,): INTETAIN 2014, LNICST 136, pp. 22-24.

Wilensky, U., & Reisman, K. (2006). Thinking like a wolf, a sheep, or a firefly: Learning biology through constructing and testing computational theories – an embodied modeling approach. *Cognition and Instruction, 24*(2), 171-209.

Wing, J. M. (2006). Computational thinking. *Communications of the ACM. 49*(3)*,* 33-35.

Wing. J. M. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society A, 366,* 3717-3725.

Wood, M. B., Turner, E. E., Civil, M., & Eli, J. A. (Eds.). (2016). *Proceedings of the 38th annual meeting of the North American Chapter of the International Group for the Psychology of Mathematics Education*. Tucson, AZ: The University of Arizona.