Evolving a K-12 Curriculum for Integrating Computer Science into Mathematics

Kathi Fisler
kfisler@cs.brown.edu
Brown University/Bootstrap
Providence, RI, USA

Emmanuel Schanzer
schanzer@bootstrapworld.org
Bootstrap/Brown University
Providence, RI, USA

Steve Weimar
sweimar@21pstem.org
21PSTEM
Conshohocken, PA, USA

Annie Fetter
afetter@21pstem.org
21PSTEM
Conshohocken, PA, USA

K. Ann Renninger
krennin1@swarthmore.edu
Swarthmore College
Swarthmore, PA, USA

Shriram Krishnamurthi
sk@cs.brown.edu
Brown University/Bootstrap
Providence, RI, USA

Joe Gibbs Politz
jpolitz@eng.ucsd.edu
Univ. of California, San Diego
San Diego, CA, USA

Benjamin Lerner
blerner@ccs.neu.edu
Northeastern University
Boston, MA, USA

Jennifer Poole
jen@bootstrapworld.org
Bootstrap/Brown University
Providence, RI, USA

Christine Koerner
christine.koerner@sde.ok.gov
Oklahoma Department of Education
Oklahoma City, OK, USA

ABSTRACT

Integrating computing into other subjects promises to address many challenges to offering standalone CS courses in K-12 contexts. Integrated curricula must be designed carefully, however, to both meet learning objectives of the host discipline and to gain traction with teachers. We describe the multi-year evolution of Bootstrap, a curriculum for integrating computing into middle- and high-school mathematics. We discuss the initial design and the various modifications we have made over the years to better support math instruction, leading to our goal of using integrated curricula to cover standards in both math and CS. We provide advice for others aiming for integration and raise questions for CS educators about how we might better support learning in other disciplines.

CCS CONCEPTS

• **Social and professional topics → K-12 education**.

KEYWORDS

Integrating computing and math education; K-12 computing education; Professional development; K-12 standards

SIGCSE '21, March 13–20, 2021, Virtual Event, USA

ACM ISBN 978-1-4503-8062-1/21/03…$15.00
https://doi.org/10.1145/3408877.3432546

ACM Reference Format:
Kathi Fisler, Emmanuel Schanzer, Steve Weimar, Annie Fetter, K. Ann Renninger, Shriram Krishnamurthi, Joe Gibbs Politz, Benjamin Lerner, Jennifer Poole, and Christine Koerner. 2021. Evolving a K-12 Curriculum for Integrating Computer Science into Mathematics. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education (SIGCSE '21), March 13–20, 2021, Virtual Event, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3408877.3432546

1 INTRODUCTION

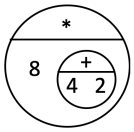
Bootstrap:Algebra [3] (BS:A) is our established project for integrating introductory computing content into middle- and high-school mathematics classes. The curriculum and its approach to professional development (henceforth PD) have evolved significantly over the years as we have learned from teachers, partners, and evaluators. In particular, our understanding of how math and computing can reinforce one another has evolved. As other projects try to integrate math and computing, we believe our observations about math teachers, math teaching, framing of CS, and key touchpoints could be useful, even in fields other than math.

Our evolution occurred on many fronts in parallel. We describe our initial assumptions about how to integrate math and CS, then describe various observations or ideas that have caused shifts in our understanding. We summarize our key lessons throughout, while also raising considerations regarding choices of programming languages and differences in USA educational standards across disciplines. The changes and our findings have emerged from a partnership between the BS:A team (authors Fisler, Schanzer, Krishnamurthi, Politz, Lerner, and Poole), experts in mathematics learning and professional development (Weimar, Fetter, and Renninger), and the Division of Secondary Mathematics Education for the State of Oklahoma (Koerner).

2 THE INITIAL EFFORT

BS:A was designed to leverage computing to help 7th-10th grade students learn to solve word problems and understand function composition in algebra [10]. Three design elements were (and still are) at the heart of the curriculum:

(1) **Circles of Evaluation:** A visual representation for (arithmetic and computational) expressions based on nested circles. Writing expressions as circles separates parsing them (identifying their structure) from computing their values. It also reinforces that expressions, like sentences, have structure.



(2) **The Design Recipe:** A step-by-step approach [9] to getting from the text of a word problem to a symbolic-form function that solves the word problem. The recipe has students work through three steps: (1) identifying the domain and range of the function described in the word problem, (2) developing concrete input/output examples of the computation to be performed for the problem, and (3) abstracting over the examples to produce a symbolic form solution to the word problem. Each step (representation) is discussed in its own right, and as part of a progression. The steps are carried out on a paper-and-pencil worksheet prior to students entering the symbolic form as code in a programming environment.

(3) **The Videogame Project:** To help students contextualize the algebraic content, students solve a series of word problems that build-up to implementing a videogame with a user-controlled player, a moving target (intersect with to earn points), and a danger (avoid to keep the game going). Game features are aligned to standard topics in (pre-)algebra, such as linear functions (to move characters), piecewise functions (to control the player via keystrokes), inequalities in the plane (to support side-scrolling), and the Pythagorean theorem (to detect collisions between the player and either the target or the danger).

Professional Development: The initial BS:A PD workshop was a three-day, hands-on, in-person event in which teachers with no prior programming experience did the curriculum as students: they designed and implemented their own videogame, working through the Circles of Evaluation and the Design Recipe. The design was intended to (a) show teachers that the game project was achievable for students new to programming, (b) help teachers gain knowledge and confidence to implement the curriculum, and (c) illustrate ways in which math and CS could align conceptually. We believed that having the teachers implement the entire game project during the three-day workshop was essential to achieving (a) and (b), while strengthening teachers' perception of (c).

The in-person PD did not spend much time discussing how to actually integrate the materials into a classroom. Facilitators (who often included experienced BS:A teachers) would discuss different high-level models (such as doing it as a standalone unit at the end of the year or doing the lessons piecemeal as they came up during the algebra course). No time, however, was allocated within the PD itself for teachers to plan their own implementations. Rather, teachers were encouraged to reach out to the PD facilitators and other teachers on the program's email list for help with planning their use of the materials.

Choice of Programming Language: The initial BS:A design used (a pure subset of) the programming language Racket [11]. In pure functional programming, functions can only take *inputs* and compute *outputs*; they cannot produce side effects (e.g., print statements) or change the values of variables (there are no assignment statements). They thus share a core semantics with algebra, enabling the alignment of math and computing in ways that conventional imperative programming does not. The team envisioned that concepts could flow freely between math and computing, without interference from the programming language.

We chose Racket in particular partly because its parenthetical syntax and placement of functions in prefix notation align particularly well with Circles of Evaluation. The Racket code for the sample circle shown in the left column is

$$(* \ 8 \ (+ \ 4 \ 2))$$

Circles can be translated to Racket by *using parentheses to mimic the circles* and placing the function at the front. We use a metaphor of an ant crawling through the expression from outside-in: when the ant "eats" into a circle, we write parentheses (to capture that circle), write the function (what appears above the line), then continue to translate the arguments (nested circles) left-to-right. This rule has proven easy for teachers and students to follow (those used to writing programs with infix operators had a harder time than did those new to programming). The uniform syntax for arithmetic operators and other (including user-defined) functions also reinforces the semantic point that students are already familiar with several functions (addition, multiplication, etc.): programming expands that known set with functions on other datatypes.

Observations

The initial design and PD model yielded several positive results. End-of-workshop evaluations (from hundreds of teachers) rated the program highly: teachers with no prior computing background created games (and were excited about it), saw connections between algebra and computing, and were eager to try it out with students. Research across multiple courses showed students making statistically-significant gains in solving word problems, both with and without the structure of the design recipe [22, 23].

However, two main concerns stood out from our internal and external assessments. During the workshop, many math teachers struggled to make deeper sense of some of the connections that we had expected and designed around. In addition, strong teacher enthusiasm after the workshop was not leading to desired adoption rates. We thus set out to understand and address these issues.

3 UNDERSTANDING FUNCTIONS

BS:A was based on an assumption that our target audience of math teachers would have a robust understanding of functions as objects (the general notion, not the programming construct) with properties (e.g., linear, injective) that captured relationships between inputs and outputs and supported re-use of computations.

In practice, we found that many math teachers had internalized an understanding of functions as equations of the form $y = mx + b$, a view reinforced by math curricula and textbooks. A teacher might thus be comfortable with the notation $y = x + 5$, but not with the notation $f(x) = x + 5$. The former was embedded in frequent (and heavily tested) exercises that contrasted 2-dimensional line graphs with symbolic forms that viewed axis names as variables. For many teachers, variable names beyond x, y, and z were initially jarring, as were descriptive function names such as *wages*, or functions that had more than one input.[1]

What is wrong with the equational framing? One major limitation of this form becomes evident when one needs to compose functions. Imagine two functions: $p(x) = 25 * x$ computes the total price of items at 25 apiece and $t(y) = y * 1.05$ augments an amount to include 5% tax. It would seem natural to compose these functions to obtain the cost of x items with tax: e.g., $t(p(10))$ computes the net price for 10 items.

Now consider the equation form. First, we might have written $p = 25 * x$ and $t = y * 1.05$, which as written cannot be composed: all the formulas have to match up their variable names (just as in BASIC programs!). So let's say we instead write $p = 25 * x$ and $t = p * 1.05$. Asked to compute the price for 10 items, the student would write

$$p = 25 * 10$$
$$p = 250$$

Substituting for p in the equation for t includes the tax:

$$t = 250 * 1.05$$
$$p = 296.5$$

In the context of middle-school mathematics, this approach may suffice. But as computations get more complex, this variable-based approach can become error-prone (for example, if someone has to do the same computation for multiple inputs, care must be taken to not use the intermediate values from a different computation): this is one reason why computing has functions, not only variables. The equation form also obscures the idea of *dependency*, and how dependencies can cascade (as in the nested composition to compute prices with tax).

More broadly, the idea that functions are just forms of equations that one calculates with is a limited, and over-generalized, student conception that can begin to form in middle school math [13, 20]. For example, as they learn to graph functions, students are taught to distinguish functions from other mathematical relations using the "vertical line test", but this does not extend to polar coordinates or to certain cases where functions on cartesian coordinates are inverted by swapping domain and range [20]. In general, functions get more sophisticated as students proceed through high-school and collegiate mathematics (consider regression models or calculus, for example). They are already more sophisticated in early computing, as functions have domains other than numbers. A limited procedural conception of functions therefore has implications for both mathematics and computing education. Computing offers a grade-appropriate and compelling context for learning a richer concept of functions for the long-term benefit of both disciplines.

[1]These comments are in no way meant to be derogatory of teachers. Their perspectives are cultural, embedded throughout the materials, texts, and conversations that many are accustomed to around teaching mathematics.

Integration Lesson 1. *Check your assumptions about how teachers understand aspects of the host discipline that you depend on in your approach to teaching CS.*

3.1 Focusing on Concepts of Functions in PD

As we looked at what backgrounds teachers were bringing to PD, we refined what we felt teachers needed to know about functions prior to the workshop. In particular, they needed a robust concept of function that included (a) seeing functions as objects with properties, rather than just an alternative notation for equations, (b) realizing that functions were not limited to single or numeric inputs and single numeric outputs, and (c) being comfortable with functions as capturing computations and models.

Knowing that these concepts may take some time to develop (and that PD time was limited), we extended the PD to include an online preparatory module on functions that would activate and, where needed, perturb existing conceptions (while also serving as an initial community-building exercise). The expected time commitment was one hour per week for each of the 5 weeks preceding the in-person workshop. All work was done in the Discourse platform [7]. We assigned teachers a series of exercises and written reflections about the nature of functions, and had them comment on each others' answers. The specific exercises were designed to introduce experiences with mathematical functions that were a little different from their textbook exercises. The exercises would support a continuous reflection on the concept of function and the way participants teach it. We hoped this would extend participants' understandings of functions, thus enabling the programming component to strengthen this concept even further. Samples of these exercises include:

(1) Asking teachers to define the concept of "function" and why it is an important concept. (We revisit this question in later instruments to see how teachers' definitions evolve.)
(2) The *Dynagraphs* activity [8] uses dynamic math software to explore a novel interactive representation of functions as single-valued mappings and connect those to algebraic expressions.
(3) Having teachers read and discuss a chapter from NCTM's "Putting Essential Understanding of Functions into Practice in Grades 9–12" [20] that raises common issues that arise in student conceptions of functions.
(4) The *FluData* activity gives participants two tables, one about the number of students infected each day with the flu and another table about the number of tissue boxes sold depending on the number of students infected with flu. Participants are encouraged to think about relationships across a set as well as the need for function composition.
(5) The *Graphing Stories* activity focuses on representing action in stories through graphs of functions. This exercise runs within Desmos [6], a graphing calculator (and more) that is popular among math teachers.

3.2 Observations

In post-workshop evaluations, many teachers reported that the pre-workshop changed their thinking about functions in mathematics. They found useful activities to bring to their own classrooms from

both the math and the CS perspectives. However, they still hadn't yet made deeper conceptual links between the Discourse activities and computing. In particular, it appeared the teachers mostly saw the videogame project as a cool application of topics that their courses covered, but not as a mechanism for learning something new about functions. At the end of the in-person PD, teachers were asking us to help them better understand what CS actually is, how it connects to mathematics, and what its standards ask of students. While we had made progress on relaxing teachers' assumptions about functions, we had not yet made the changes needed to enable the teachers to *transfer* [5, 25] concepts about functions between math and CS.

Integration Lesson 2. *Making sure that participants have the foundations to connect two disciplines is necessary but not sufficient when designing PD. Integration efforts must also include explicit opportunities for teachers to articulate connections between the disciplines. Both parts are needed to achieve significant transfer.*

4 INCLUDING MATH TEACHING PRACTICES

The original BS:A design made extensive use of paper-and-pencil worksheets for both Circles of Evaluation and the Design Recipe (as well as other intermediate exercises). In an initial evaluation of the materials [24] with a mix of math teachers and CS teachers, all of the math teachers commented positively on the familiar use of worksheets, while the CS teachers found the structure unusual, if not outright foreign. Over the years, many math teachers have commented on BS:A's familiar pedagogic tools being part of what made them comfortable with the program.

The Discourse activities, which were designed by experts in online math learning (co-authors Weimar and Fetter from the former Math Forum), introduced another well-regarded practice called "Notice and Wonder" [16]. In this practice, students are initially given a description of a scenario or an artifact (such as a graph or diagram), but not a specific question to answer about it. Students instead respond to two prompts: *"What do you notice?"* and *"What do you wonder?"*. Without the pressure to provide a correct answer, students' observations center around making sense of the scenario or artifact, attending to interesting features or patterns, subparts and their relationships, the context, and so on. The protocol invites participation (since there is minimal risk of saying something "wrong" and all observations are deemed worthy of expression). It centers students' thinking while also helping them attend to the key structural features of the scenario (one of the practices highlighted in many K-12 math standards). This protocol is not limited to students: for example, it is the center of a weekly New York Times series called "What's Going on in This Graph?" [14], also developed with the support of co-author Fetter. The BS:A staff revised the curricular materials to introduce Notice-and-Wonder activities into the computing portions of the curriculum, hoping to draw teachers' attention to such concepts as re-use, syntax, and the structure of expressions in programming.

We have also begun to adapt a lesson structure from the math education community known as Math Workshop [12]. Each lesson now has three segments called *Launch*, *Investigate*, and *Synthesize*. The first engages students in sensemaking and exploration of the mathematical setting for a problem, the second covers specific techniques for working on a problem, and the last engages in reflective activities that can consolidate learning and generalizations. Exercise structures like these are common in math teaching, but less so in computer science.

Computing education as a discipline may have a lot to learn from these methods. Our point here, however, is that independent of their standalone impact on CS learning, embracing task structures that are established in math education can help math teachers accept and frame CS content.

Integration Lesson 3. *Understand teaching practices in the host classes and adapt your content to them (when feasible), rather than expect teachers to adopt CS teaching practices while also learning new CS content.*

5 UNDERSTANDING TEACHERS

Until this point in the development of BS:A, our feedback on the PD and the curricular materials came through two main sources: (1) end of workshop surveys and (2) two rounds of external evaluation focused on teachers' experiences with both PD and using the materials. Soon after we added the Discourse-based preparatory module, co-author Ann Renninger, an educational psychologist, began to conduct a deeper exploration of how participants' understanding of math, CS, and their potential integration was evolving through the various PD activities.

How teachers' thinking evolves has design implications for PD. Other studies of mathematics PD have observed a correlation between participants' reasons for attending PD and the kinds of PD activities that motivate them to go deeper into material [18]. At some level, this is intuitive: a teacher who has been instructed to attend PD but doesn't really see how the content is relevant to them will engage with PD differently than a participant who already has the background and interest to teach the content addressed in the workshop. The prior research is, however, more nuanced. Studies have shown that it is possible to support learners to develop their interest, and that this is positively impacted when the assistance or instruction that is provided is responsive to their present understanding [17, 19]. For some teachers, that means engaging them in supporting fellow participants. Others need help making connections between new content and their existing teaching practices. If the prior findings held in our integrated setting as well, we could draw on past literature to further adapt the design of our PD.

For the past two and a half years, we have been conducting a mixed methods study with participants in our workshops in Oklahoma (starting with the first group that worked with the online activities described in section 3.1). Our data thus far suggest that prior exposure to CS is not a strong factor in whether teachers go on to use our materials. Having CS background affects teachers' initial comfort with writing code, but their interest in mathematics and understanding integration is what's critical. In particular, *teachers who adopt materials are driven by their interest in math, rather than an independent interest in CS.* The success of PD at reaching teachers ultimately depends on how the teachers view mathematics, and whether the PD provides time and space for them both to understand the mechanisms of integration and to develop core skills in CS.

Integration Lesson 4. *Teachers who are interested in learning to teach CS start with their understanding of their home discipline, but aren't necessarily aware of what CS means in that discipline. Continuing to engage teachers in thinking about learning of their home discipline may be critical for motivating them to include CS content, even when they believe CS content is valuable for students.*

From the perspective of workshop design, the key takeaway is that teachers need to be exposed to more than the content that they will eventually teach to students. They need to be exposed to content that will promote their own (sometimes professional, sometimes personal) interest in integrating computing into their home discipline. The workshop design needs to stretch the possibilities they see for their teaching of their home discipline.

6 GO WIDE BEFORE GOING DEEP

Early in the development of BS:A, the team would hear from teachers that they had "adopted" the material, when in fact they were using only the earliest lessons (on Circles of Evaluation and perhaps creating images). Before we started to study teachers' development via PD (section 5), we viewed such "adoptions" largely as failures. While such adoption still falls short of what we hope to achieve in the long term, we now see this form as a critical stepping stone. Our current challenge is to design PD that encourages teachers to keep taking these small steps as they build up their understanding of computing and how it integrates with math.

Our original PD design (section 2) focused on getting participants to implement the videogame project within the 3-day in-person workshop. We felt the game was important for two reasons: covering multiple math topics and engaging students. In open-ended evaluation questions post-PD, teachers often mentioned that they were proud for having created a game themselves, and that they thought their students would enjoy the game project. But our study data from the Oklahoma teachers suggests that at least for some teachers, the game project goes too deep too fast. It is too big of a bite for easy integration, and was intimidating teachers more than inspiring them. Instead, it appeared we would be better off exposing teachers to multiple initial points of contact between math and CS, and introducing the game project over a longer period of time.

In our Oklahoma workshops, we have stopped getting to the game project in the 3-day PD. Instead we introduce three forms of early content: (day 1) Circles of Evaluation and composing images; (day 2) writing linear functions and using them in simulations; and (day 3) manipulating data tables to create data subsets and plots from Bootstrap:Data Science [4]. Some teachers first integrate the content from day 1, while others start with the content from day 3 (based on what makes the most sense for their classes). The end of day 2 looks ahead to where the materials might go for creating games and simulations, but without conveying the expectation that teachers are ready to implement this for themselves.

Since making this change, we find that roughly half of each starting cohort (roughly 24 teachers each time) has asked to return for more PD. Many seek to redo the first PD to solidify their knowledge, while others have gained confidence and are ready for more advanced material. In the coming academic year, we will be introducing a second-level workshop for those who have already used the early content with students: this workshop will introduce more of modeling and simulation that could be used to reach the videogame project or to work more deeply with applications to data science (the participants will decide).

Integration Lesson 5. *A PD model designed around a culminating activity that will be engaging to students may be harder for teachers to adopt than one that helps teachers make incremental steps towards incorporating CS into their classes. Teachers may need or want to repeat the content of their first PD before moving on to learning additional material.*

7 MAKING MATERIALS MORE FLEXIBLE

When BS:A started (over a decade ago), it was designed as an afterschool program [21] taught by volunteers with programming experience but no classroom-management experience. The initial curricular materials scripted content and classroom-management techniques into neat 90-minute segments that culminated in finishing the videogame project. Once our focus shifted to working with in-school math teachers, we reworked the materials to take out classroom-management and to include instructions on teaching computing (for teachers who lacked programming experience). We also rebranded the 90-minute segments as units focused on a specific content topic. The focus on the videogame remained. (This rewrite occurred years before the modifications to PD described in sections 3.1 through 5.)

The videogame-based materials supported teachers who wanted to use them as a whole unit at the end of the school year (using the game project to keep students engaged as summer neared), but required a heavy lift from those who wanted to cover the material integrated in pieces across a longer span. Those teachers had to subdivide the materials for themselves. For teachers who wanted to integrate computing but not do the game project, the task was even more challenging. These teachers needed curricular materials that separated the underlying math-aligned computing concepts from the videogame project. In addition, while some teachers needed scripted materials (as they were getting started), others needed lesson outlines that they could readily tailor to their own classes.

The BS:A curriculum has since been rewritten once more, this time as a series of individual lessons that can be remixed in diverse contexts. We present the curriculum as different "pathways" that build upon shared lessons while leading to different end goals. The lessons are no longer minutely scripted, but instead provide tasks structured into the *Launch*, *Investigate*, and *Synthesize* segments described in section 4. Lessons are now downloadable as Google Docs, to make it easier for teachers to create their own lesson plans around our notes. Both of these steps acknowledge that teachers need to find their own ways to integrate content, and those paths may differ considerably, even (as we have found) for teachers in the same district or demographic community.

8 REVISITING PROGRAMMING LANGUAGES

Over the years, we have had many internal discussions about which programming language to use. Even within functional programming, we have debated the tradeoffs between parenthetical syntax with prefix notation and a more traditional syntax (in both math and CS) that uses infix for operators and prefix for functions. At a high-level, the tradeoff has been between the ease of translating

from Circles of Evaluation to parenthetical code on one side and the familiarity of infix syntax for arithmetic and alignment with mainstream programming syntax on the other.

When we started BS:A, there were no K-12 CS standards and few large-scale efforts to teach CS in USA secondary schools. That landscape has changed dramatically. Many districts now consider using BS:A within a multi-year progression leading to standalone CS courses in later school years. Some teachers want to use BS:A as part of CS courses that combine material from multiple providers. Even our own integrated content has expanded considerably, as we have introduced content on data science and modeling into our core program (section 6). Responding to this context, we have shifted our materials to use Pyret [15], a different functional-programming language with Python-like syntax (designed by authors Krishnamurthi, Lerner, and Politz). This has opened up considerable possibilities for connecting our materials to other efforts.

From a K-12 CS perspective, a natural question is whether shifting to blocks (rather than textual programming) would have been better. Our data do not suggest that blocks would make a significant difference in the context of integrating into math classes. First, our Circles of Evaluation are already a diagrammatic syntax (that matches the expression structure of functional programming). Second, our syntactic needs are lighter-weight than in many early programming projects due to our use of functional programming: our students need only expressions, function calls, function definitions, and conditionals. In terms of syntax errors (which blocks can help avoid), the main issues we observe in practice are in getting the correct order of arguments and in remembering to close parentheses after function calls. Finally, blocks would miss a key opportunity from integrating computing into math, where students need to be comfortable working in textual symbolic form. Many teachers actually appreciate that error messages encourage precision!

9 THINKING ABOUT STANDARDS

When BS:A started, we were thinking about how to leverage computing to help students contextualize mathematics. We wanted the videogame project to help students see why math is useful, in hopes of encouraging them to stay engaged with math through middle- and early high-school (when many students turn away from math due to the challenges of algebra). Over the years, as we have worked with more math teachers, as well as talked to more school districts (where impact on state testing scores in math is a primary concern), we have had to align our materials to math standards. With the growing adoption of CS standards for K-12, some of our partners (including Oklahoma) are asking how to use our approach to integration to satisfy some of their CS standards as well. We are thus now at a point of asking how to *simultaneously* satisfy math and CS K-12 standards using our materials.

Current CS [2] and math [1] standards have interesting differences, many of which center around differing notions of variables and control flow. Elementary school CS standards introduce the idea of stateful variables; in math standards, variables arise in middle-school and represent different values over time, but not stateful ones. CS standards emphasize loops for control flow, whereas function composition represents control flow in math. Both math and CS share core ideas of conditional and repeated computation, but CS frames these concepts through an emphasis on imperative programming. In contrast, math might introduce these conceptions through models based on piecewise functions or models that are repeatedly applied in the context of simulation. In short, many of the content topics in the algorithms, programming, or computational thinking portions of CS standards *could* align with a math context, but not necessarily using the specific implementation decisions, phrasings, or grade-band progressions that have been hard-coded into CS standards. Taking teachers' needs seriously (section 5), the CS side may need to be the ones to bend.[2]

Comparing the K-12 Practice standards for math and CS is even more revealing. Math practice standards tend to focus on cognitive and behavioral aspects of *solving mathematical problems*, while the CS practice standards focus more on *creating artifacts*. While the contrasts do reflect genuine differences between the disciplines (CS is not a sub-field of mathematics, after all!), we find the math practices to be much more useful in thinking about how to align content in integration. A (math) practice like "look for and make use of structure", for example, applies in both disciplines, but is not articulated within CS standards. "Make sense of problems and persevere in solving them" (math) is important to both fields as well, but there is no analog in CS. CS standards emphasize working collaboratively and producing artifacts; neither of these hint at the sort of mental or executive-function behaviors that are perhaps a more useful guide for integrating learning across disciplines.

Integration Lesson 6. *Those developing or using integrated curricula should not expect that standards across the disciplines will align as written, even if the underlying spirit of the content and practices are fundamentally similar. This can pose a challenge when creating materials that will be accepted by schools and districts.*

10 REFLECTION AND CONCLUSION

This paper is a retrospective experience report, looking at the evolving design decisions and lessons learned over the past 6 years of trying to create curricular materials for integrating CS into 7th-10th grade math classes in the USA. We began this project with what we thought was a clear understanding of where math and CS could meaningfully align in the context of middle-grades math curricula. As we have paid more attention to pedagogic content knowledge, cultures and practices of math teaching, and how to engage teachers over time (as informed by feedback from and interviews with teachers), we have come to understand integration differently. We appreciate the fine-grained stages in which such work might take hold, as well as the preparation and long-term support needed for teachers to work effectively with integrated materials. The lessons stated throughout this paper summarize our core insights.

Arguably, the lessons we raise are not new. Instructional-design practices such as knowing your audience and the need to explicitly teach for transfer are well established. The challenge, of course, lies in the practical application of such results. We needed to work closely with our teachers to understand what we had taken for granted and what we needed to teach explicitly, even after (thinking we were) applying these lessons to our initial design. We hope that highlighting what we have learned will help other programs explore relevant concrete issues earlier in their design processes.

[2]This view reflects co-author Fisler's experience writing CS standards in three states.

ACKNOWLEDGEMENTS

We are deeply grateful to the hundreds of teachers and thousands of students who have used our curricula and provided feedback. We thank our external evaluators, McClanahan and Associates. We have been supported by the US National Science Foundation (grants 1535276,1738598,1738606,1829544), Microsoft, Google, Jane Street Capital, Code.org, and TripAdvisor; and numerous school districts. We thank them all.

REFERENCES

[1] 2010. Common Core Standards for Mathematics. http://www.corestandards.org/Math/; retrieved August 14, 2019.
[2] 2017. The CSTA Standards. https://www.csteachers.org/page/standards, last accessed 2018-05-04.
[3] Bootstrap Algebra [n.d.]. Bootstrap:Algebra Curriculum. https://www.bootstrapworld.org/materials/algebra/.
[4] Bootstrap Data Science [n.d.]. Bootstrap:Data Science Curriculum. https://www.bootstrapworld.org/materials/data-science/.
[5] J.D. Bransford and D. Schwartz. 1999. Rethinking transfer: A simple proposal with multiple implications. In *Review of Research in Education*. Vol. 24. American Educational Research Association, 61–100.
[6] Desmos. [n.d.]. https://www.desmos.com/.
[7] Discourse. [n.d.]. http://www.discourse.org/.
[8] Dynagraphs. [n.d.]. http://files.mathematicalthinking.org/wsp/dynagraphs/.
[9] Matthias Felleisen, Robert Bruce Findler, Matthew Flatt, and Shriram Krishnamurthi. 2001. *How to Design Programs*. MIT Press. http://www.htdp.org/
[10] Matthias Felleisen and Shriram Krishnamurthi. 2009. Why Computer Science Doesn't Matter. *Commun. ACM* 52, 7 (July 2009).
[11] Matthew Flatt and PLT. 2010. *Reference: Racket*. Technical Report PLT-TR2010-1. PLT Inc. http://racket-lang.org/tr1/.
[12] Wendy Ward Hoffer. 2012. *Minds on Mathematics: Using Math Workshop to Develop Deep Understanding in Grades 4-8*. Heineman.
[13] C. Kieran. 2007. Learning and teaching algebra at the middle school through college levels. In *Second handbook of research on mathematics teaching and learning*, F. K. Lester (Ed.). Vol. 2. National Council of Teachers of Mathematics, Information Age Publishing, Charlotte, NC, 707—-762.
[14] New York Times Learning Network. 2017. Announcing a New Monthly Feature: What's Going On in This Graph? https://www.nytimes.com/2017/09/06/learning/announcing-a-new-monthly-feature-whats-going-on-in-this-graph.html.
[15] Pyret [n.d.]. The Pyret Language and Programming Platform. https://pyret.org/.
[16] Max Ray. 2013. *Powerful problem solving: Activities for sense making with the mathematical practices*. Heinemann.
[17] K.A. Renninger and S.E. Hidi. 2020. To Level the Playing Field, Develop Interest. *Policy Insights from the Behavioral and Brain Sciences* 1 (2020).
[18] K. A. Renninger, M. Cai, M. Lewis, M. Adams, and K. Ernst. 2011. Motivation and learning in an online, unmoderated, mathematics workshop for teachers. *Special Issue: Motivation and New Media. Educational Technology, Research and Development* 59, 2 (2011), 229–247.
[19] K. A. Renninger and S. E. Hidi. 2019. Interest development and learning. In *The Cambridge handbook of motivation and learning*, K.A. Renninger and S.E. Hidi (Eds.). Cambridge: Cambridge University Press, 265–296.
[20] Robert Ronau, Dan Meyer, Terry Crites, and Barbara Dougherty. 2014. *Putting Essential Understanding of Functions Into Practice in Grades 9-12*. National Council of Teachers of Mathematics.
[21] Emmanuel Schanzer, Kathi Fisler, and Shriram Krishnamurthi. 2013. Bootstrap: Going Beyond Programming in After-School Computer Science. In *SPLASH Education Symposium*.
[22] Emmanuel Schanzer, Kathi Fisler, and Shriram Krishnamurthi. 2018. Assessing Bootstrap:Algebra Students on Scaffolded and Unscaffolded Word Problems. In *ACM Technical Symposium on Computer Science Education*.
[23] Emmanuel Schanzer, Kathi Fisler, Shriram Krishnamurthi, and Matthias Felleisen. 2015. Transferring Skills at Solving Word Problems from Computing to Algebra Through Bootstrap. In *ACM Technical Symposium on Computer Science Education*.
[24] Emmanuel Tanenbaum Schanzer. 2015. *Algebraic Functions, Computer Programming, and the Challenge of Transfer*. Ph.D. Dissertation. Harvard Graduate School of Education.
[25] M. K. Singley and J. R. Anderson. 1989. *The Transfer of Cognitive Skill*. Harvard University Press.