Received: 29 July 2020 | Accepted: 2 March 2021

DOI: 10.1111/bjet.13080

ORIGINAL MANUSCRIPT

British Journal of Educational Technology BERA

Programming as a language for young children to express and explore mathematics in school

E. Paul Goldenberg[1] | Cynthia J. Carter[2]

[1]Education Development Center (EDC), Waltham, MA, USA

[2]The Rashi School, Dedham, MA, USA

Correspondence
E. Paul Goldenberg, Education Development Center (EDC), 43 Foundry Ave., Waltham, MA 02453, USA.
Email: pgoldenberg@edc.org

Funding information
National Science Foundation, Grant/Award Number: 1741792 and 1934161

Abstract

Natural language helps express mathematical thinking and contexts. Conventional mathematical notation (CMN) best suits expressions and equations. Each is essential; each also has limitations, especially for learners. Our research studies how programming can be a advantageous third language that can also help restore mathematical connections that are hidden by topic-centred curricula. Restoring opportunities for surprise and delight reclaims mathematics' creative nature. Studies of children's use of language in mathematics and their programming behaviours guide our iterative design/redesign of mathematical microworlds in which students, ages 7–11, use programming in their regular school lessons *as a language for learning mathematics*. Though driven by mathematics, not coding, the microworlds develop the programming over time so that it continues to support children's developing mathematical ideas. This paper briefly describes microworlds EDC has tested with well over 400 7-to-8-year-olds in school, and others tested (or about to be tested) with over 200 8-to-11-year-olds. Our challenge was to satisfy schools' topical orientation and fit easily within regular classroom study but use and foreshadow other mathematical learning to remove the siloes. The design/redesign research and evaluation is exploratory, without formal methodology. We are also more formally studying effects on children's learning. That ongoing study is not reported here.

KEYWORDS
activity-based learning, coding, constructionism, mathematics, primary education

Practitioner notes

What is already known

- Active learning—*doing*—supports learning.
- Collaborative learning—doing *together*—supports learning.
- Classroom discourse—focused, relevant *discussion*, not just listening—supports learning.
- Clear articulation of one's thinking, even just to oneself, helps develop that thinking.

What this paper adds

- The common languages we use for classroom mathematics—natural language for conveying the meaning and context of mathematical situations and for explaining our reasoning; and the formal (written) language of conventional mathematical notation, the symbols we use in mathematical expressions and equations—are both essential but each presents hurdles that necessitate the other. Yet, even together, they are insufficient especially for young learners.
- Programming, appropriately designed and used, can be the third language that both reduces barriers and provides the missing expressive and creative capabilities children need.
- Appropriate design for use in regular mathematics classrooms requires making key mathematical content obvious, strong and the 'driver' of the activities, and requires reducing tech 'overhead' to near zero.
- Continued usefulness across the grades requires developing children's sophistication and knowledge with the language; the powerful ways that children rapidly acquire facility with (natural) language provides guidance for ways they can learn a formal language as well.

Implications for policy and/or practice

- Mathematics teaching can take advantage of the ways children learn through experimentation and attention to the results, and of the ways children use their language brain even for mathematics.
- In particular, programming—in microworlds driven by the mathematical content, designed to minimise distraction and overhead, open to exploration and discovery *en route* to focused aims, and in which children *self*-evaluate—can allow clear articulation of thought, experimentation with immediate feedback.
- As it aids the mathematics, it also builds computational thinking and satisfies schools' increasing concerns to broaden access to ideas of computer science.

OUR STUDY

The National Science Foundation defines their early stage design and development studies to include iterative development of an intervention or instructional approach—in our case, programming microworlds through which children express and explore their classroom mathematics—testing to provide feedback in the development process, and the creation of measures to assess the implementation of the intervention. Our research addresses issues of feasibility and usability; it also gathers preliminary evidence regarding promise for improving education outcomes in elementary school mathematics. The bulk of this paper describes specific ideas we are exploring and why we focus on those particular ideas.

We use a number of formal methods—including video recording and coding, cognitive interviews with children, even comparison classrooms—to help us document what we are seeing and interpret its meaning but emphasise at the outset that our study, though with over 400 children now, is not statistical. The role of comparison classrooms is to help us understand how classrooms with similar student demographics, teacher characteristics and district contexts approach the teaching and learning the mathematics content our microworlds present. This helps us contextualise what we are seeing in our treatment classrooms—the purpose is to help us recognise contrasts on various measures including engagement and interest but not (at this point) to quantify them. Coded video gives us information about those measures, but more important (at this point) is the information about the iterative redesign of our microworlds. We work closely with our evaluation partner, Horizon Research, Inc. (HRI), to identify specifications for the classroom observation protocols, making use of existing protocols and rating scales. Before coding year 2 data, two coders will use the protocol independently until inter-rater reliability reaches 0.80 or more. After reaching reliability, checks will be conducted at regular intervals to ensure reliability is maintained.

This paper does not report final, formal results—that is neither the purpose of this phase of our research nor the purpose of this paper—but takes on the task of elucidating the *ideas* behind the study, the reason *why* our questions and results can matter to mathematics learning, and what we are currently observing in practice. We begin with three foundations of thinking and learning: doing or action, abstraction and language. We then argue that computer programming has the potential to connect and synthesise these foundations.

Doing

The notion that we learn primarily from our actions is old. The familiar 'I hear and I forget, I see and I remember, I do and I understand'—attributed (perhaps incorrectly) to Confucius—is over 2000 years old; the idea of apprenticeship for learning no doubt goes back pretty much to the origin of our species. That idea continues to echo in learning theory today. Despite the role language plays in transmitting ideas from one generation to the next—and even the theory that language developed, in part, because there are some things that *cannot* be communicated through action alone—some theoretical positions hold physical action with 'concrete' objects to be essential. Perhaps the most familiar scientifically researched position is Piaget's.

> Knowledge is derived from action…. To know an object is to act upon it and to transform it…. To know is therefore to assimilate reality into structures of transformations, and these are the structures that intelligence constructs as a direct extension of our actions. (Piaget, 1971, p. 28)

Piaget insisted that he was not a pedagogue and was not trying to advise educators (see, eg, Evans, 1973)—he was trying to understand child cognition—but arguments for the active side of learning often trace to him. And, of course, to Dewey (1933), and Bruner (1985) and, well, Confucius. Pop-culture interpretations regularly mix conservatism—it is easier to have kids sit and watch and listen than to manage truly exploratory active learning—with extremism, versions of Piaget and other theoreticians that are exaggerated beyond what the theoreticians would ever have laid claim to (eg, Clements et al., 2017; Herbel-Eisenmann et al., 2016). The resulting practices intermingle show-and-tell pedagogy with scripted 'active learning'—manipulations of objects without a clear (and defended) theory of why *those* objects at *that* time handled in *this* way with *these* formal instructions serve the learning goal of the moment (eg, Ball, 1992; Willingham, 2017). This mixed theory is sometimes accompanied by claims about 'abstract' and 'concrete' that do not really match the realities we see in children (Baroody, 2017; Sarama & Clements, 2016).

There cannot be *any* doubt—none at all—that *something* about 'do' having priority over 'hear' must be true: that notion has just lived too long and resurfaced in too many forms to be dismissible. And the notion that 'concrete' precedes 'abstract'—in some sense—also recurs in enough forms to be convincing without yet another piece of research.

But we also know that both of these statements must be tempered.

For one thing, we are aware that we (and children and babies) learn *also* just from watching and listening. Also, we know that people who, from birth, have been so severely motorically disabled that they cannot move about independently or manipulate objects or even speak may well be *fully* capable learners, given rich enough experiences seeing and hearing, and may be able to express what they have learned (and learn more) given communication tools that use whatever control they do have (Goldenberg, 1979). So, how *does* doing acquire its status over hearing and seeing? Perhaps because doing gives us a richer soup of experiences. What children *do*, they typically also see and chatter about.

Abstractions

Children, even babies, deal in abstractions. Language is an abstraction. The 18-month-old who learns 'doggie' abstracted it from experiences with wiggling, wagging, woofing, jumping and licking, and from people making an extremely varied and chaotic range of noises that all include the word doggie. How, exactly, the child assigns 'doggie' to the *animal* and not, say, to the animal's actions is less clear, and the precision of assignment is also not great—a cat or squirrel might elicit the 18-month-old's accusatory pointer finger and the exclamation 'doggie!'—but they are *constantly* abstracting. The word 'the', acquired early, is super-abstract—nothing even to point to. Pre-schoolers' and kindergarteners' drawings of people are not what their eyes see but a representation of features that are essential for the current purpose—people have faces and arms and legs and fingers—with everything else stripped out. That is abstraction. Abstractions remove details—'doggie' is general for all children (Sperry et al., 2019), not just *that* doggie doing *that* barking and wagging—and if we *start* with abstractions in school, some of the missing details remove so much context that the meaning is lost. The notion of a 'rich soup' is gone. A statement like 2 + 5 = 7 is quite *concrete* to us as adults—we already have rich associations to each of those five printed characters (Sarama & Clements, 2016)—but seven is not initially more meaningful to a child than *perrito* is to the English speaker (or *puppy* is to the Spanish speaker). The child can deal with abstractions, but things that have *no* meaning are *not* abstractions; they are mere noise, just as '+' may be a mere scribble (cf. Dewey, 1933).

It is important to keep in mind that *language* we use with others *is also action*, and the reactions we get are feedback on that action. Language expressed to a computer is also an action that gets a reaction. In that sense, interactive language can be very 'concrete'.

Because it is also abstract, a mental representation, it may be able to support thinking in ways that manipulation, by itself, would not.

Natural languages

Natural languages—all of them, spoken or signed—are optimised for communicative speed and ease, not precision. No surprise. Natural language evolved with in-person use—not written, not even over Zoom. In-person use has features—eg, context, vocal and visual backchannel responses that signal understanding (or lack thereof), people's immediate presence for clarification, and redundancy—that make imprecision tolerable and error-correction easy. (Besides, a total lack of ambiguity is unachievable, given the unlimited and changing context of natural communication.) Communication failures occur but the success rate is remarkably high (cf. Gleitman & Papafragou, 2013). Written forms, for natural languages that have them, are generally less streamlined[1] and add details (eg, punctuation) to help make up for lack of vocal tone and tempo and the absence of the speaker, but they, too, depend on redundancy to support 'error correction', allowing reasonably reliable communication despite noise. That redundancy is what makes kindergarteners' writing intelligible despite gross misspellings; it may require thought but it remains interpretable with the slips identifiable.

Formal languages

Natural language depends on redundancy to survive its informal and unpredictable environment and use. Formal languages like conventional mathematical notation (CMN) and computer languages (or musical notation) deliberately eliminate redundancy to optimise for precision and concision. The extreme concision and precision of CMN is what makes it so perfect for *doing* mathematics but so unforgiving for *learning* it—what Scott Kim (personal communication) referred to as 'not a good user interface for [early] learning'. That concision means that minor slips—the kind that we hardly even notice in speech and that we can mostly decipher in even very garbled writing—render a computer statement or a mathematical expression or equation simply wrong with little clue as to which part is in error. Which character in 12 + 5 = 7 is in error? There are several candidates and no way to tell.

A hybrid

Computer languages are designed to be concise and precise. Unlike mathematical notation on paper, they are *live*, allowing interaction with feedback.[2] This hybrid of precision and supporting error correction (through interaction) is what motivated us to explore the effectiveness of having children use programming to do and learn mathematics. More will be said about it later.

WRITTEN LANGUAGE AND MATHEMATICS

Typically, children acquire a large part of their (common-use, non-technical/academic) adult vocabulary and most of their language's grammar (ignoring production errors and not including conscious grammatical analysis) by the time they are six or earlier (Fletcher & MacWhinney, 1995). Reading and writing are acquired *after* facility with the language itself, requiring time and effort and remaining spotty for far more than another 6 years.

Children enter school after years of casual communication. Instruction in the written representation of natural language is built on that rich foundation. By contrast, mathematics instruction in school often *begins* with written representations and keeps them central to each new conceptual move. But children do not bring to these written representations in mathematics the same foundational understanding they bring to the writing of natural language. Attempting to use written mathematics to *provide* a foundation for its ideas is a fundamentally different task from teaching children to write prose.

Because children are typically taught to write mathematics at the same time that they are taught to write their natural language, teachers often teach the writing similarly, but there are fundamental differences between them beyond even the enormous difference in foundational knowledge.

First, reading *any* natural language text is different from nearly every other kind of visual processing. The ability to recognise familiar objects in unfamiliar orientations is not born in, but infant brains quickly develop such essential geometric transformations, letting babies recognise a bottle even when the nipple does not face them directly. This visual skill takes learning but becomes automatic. For reading and writing, children must suppress that hard-won, now-automatic skill; otherwise p, d, b and q are all the same. Writing ɛ and ɜ interchangeably is normal, totally expectable at first. Distinguishing objects based on their orientation is a reading-writing-only exception. Written forms for natural language and mathematics also treat order differently. We rarely attend to order of three objects we *see*, but *reading* requires it: *was* and *saw* are different. *Mathematical* print changes the rules yet again. While 315 and 513 are unequal, 3 + 5 and 5 + 3 are equal.[3] And unlike prose text, mathematical expressions are not read strictly left to right.[4] Convention requires reading/processing both $5+4\times 2$ and $5\times(4+2)$ right to left, a strain to teach and to learn.

And, unlike prose text, mathematical notations are two-dimensional, requiring attention to both vertical and horizontal position. Students often encounter that first with graphs and charts. Later, they see it even in CMN. The typically uneven size and level of children's scrawls are just aesthetic concerns in prose writing; meaning is not affected. But in mathematics, 315, 31^5 and 31_5 mean different things. Young children do not use those notations, but they do see $\frac{1}{2}$ early. By middle school all these distinctions matter.

Using the language brain

We believe that not enough attention is paid in mathematics education to the way children can use their spectacular facility with natural language—*language itself*, not its written representation—in learning mathematics. Some of those linguistic strategies, and ways that math educators can tap them, are reported in Goldenberg and Carter (2020).

Spoken[5], 'five-eighths plus five-eighths' generally evokes the correct 'ten-eighths' response. Kindergarteners who know nothing about eighths are still quite sure that five of them plus another five of them are ten of them (see, eg, Goldenberg & Carter, 2020; Goldenberg et al., 2017). Without that (unwritten) understanding *first*, without having built a *logic* for fractions, 9-year-olds who encounter written expressions like $\frac{5}{8}+\frac{5}{8}$ are inclined to interpret the +sign as meaning 'add everything in sight' and make the canonical error. Similarly, some mathematical properties (like distributivity) seem essentially built into early cognition (see, eg, Goldenberg et al., 2012). Yet, when that idea and its name are 'introduced' to 8-year olds (as commonly mandated in the US), it is typically *taught* through a written string like $8\times 7=8\times(5+2)=(8\times 5)+(8\times 2)=40+16=56$. Far from adding precision or a new idea, this obfuscates what children already know. Processing such a string takes focus and effort, so it is not the optimal way to *introduce* ideas to an 8-year-old. In fact, despite your own mathematical fluency and adult cognition, probably even you zipped through it without reading closely enough to see if we typed it correctly. For a beginner, the cognitive load of decoding the notation obscures the idea.

We take CMN for granted, but it was a relatively recent invention on which mathematical progress absolutely depended. The features that make it so valuable to mathematics are its concision and precision. But, for learners, this comes at a price. Concision removes redundancy. In speech and prose text, some redundancy *helps* understanding. Context helps a beginning reader recognise a word or infer its meaning. And in speech and prose text, minor imprecisions rarely affect comprehension, which is on reason it is so difficult to proofread a document and catch all the errors.[6] In CMN, one wrong character obscures the meaning.

Pre-schooler's letter-salads show only a recognition that letters tell stories. Primary teachers understand that invented spelling—technically 'incorrect'—is evidence of progress, the emerging correct sense that letters encode specific speech *sounds* (Guo et al., 2018). Conventional spelling comes later. (We similarly accept 'firemans' without instant correction; the meaning is clear, cosmetic details will come later.) Children learn to read and write text by applying a reservoir of world experience to the thought, content and meaning of spoken language that the text records.

In mathematics, too, thought, content and meaning must come first, conventions later. The difficulty in *acquiring* CMN is that, for young students, world experience and natural language do not sufficiently supplement CMN's concision to support that learning; the difficulty of *using* CMN makes it not ideal for building a basis of thought, content and meaning for mathematics. When CMN does get taught, teachers must allow the same level of 'algebraic babbling' (Cusi et al., 2011) that is natural and necessary in developing spoken and written natural language.

Here is the problem. On the one hand, in order to express and explore mathematical ideas, even very young children need greater precision than natural language provides. Imaginably, that could be the role of CMN. But learning *any* subject—especially mathematics, which requires focused control of attention—via a language (eg, CMN) that one is not fluent in raises the cognitive load.

PROGRAMMING AS A LANGUAGE FOR MATHEMATICS

Because computers are relatively new for schools, programming for children is, too. Seminal works (eg, Papert, 1972, 1980) explored what was *possible*, deliberately ignoring constraints imposed by formal school settings. But computers are now widespread, and 'coding' is the new term, with new initiatives, standards and Olympiad-like challenges in many countries[7]. Despite that—and perhaps because traditional school content is even more regulated with its own standards and frameworks today—coding tends to appear, within schools, as the province of explicit robotics or coding classes, not well connected with other school learning. Except in formal courses, people do not write code in order to use loops! Nor do they learn loops just to program for programming's sake. People program to *make* things. Programming languages are designed to *help* us solve problems. Could not programming then be seen as an adjunct to, not a departure from, other school learning? That is how we see it in connection with mathematics, where the available languages—natural language and CMN—do not quite suffice. Spoken language can be used far more powerfully for learning than is common, but it is inadequate for *doing* mathematics; CMN is perfect for doing, but rough for learning.

Our own thinking about using programming to learn and teach mathematics has roots in efforts aimed at functioning within constraints of expectable school settings (especially in Benton et al., 2016, 2017; Noss & Hoyles, 2018; Sendova, 2013; Sendova & Sendov, 1994). By tailoring the environment appropriately and by using our understanding of how children's

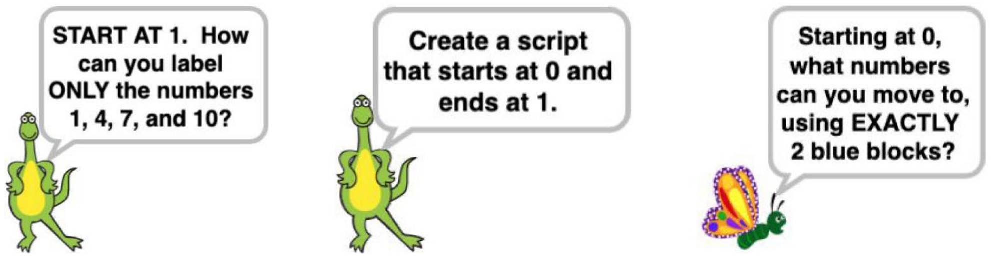
START AT 1. How can you label ONLY the numbers 1, 4, 7, and 10?
Create a script that starts at 0 and ends at 1.
Starting at 0, what numbers can you move to, using EXACTLY 2 blue blocks?

FIGURE 1 Puzzles with smaller numbers [Colour figure can be viewed at wileyonlinelibrary.com]

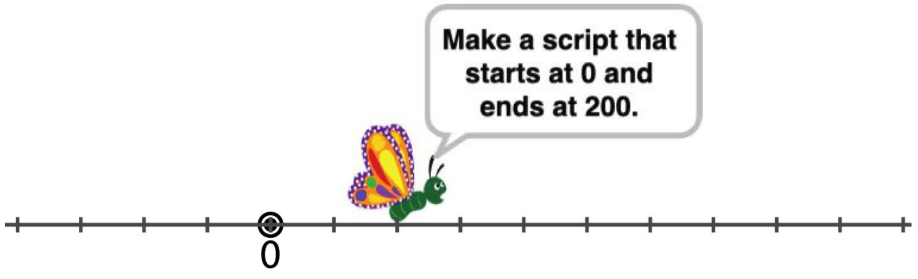

FIGURE 2 Puzzles can feel mathematically advanced and exciting by using children's linguistic 'ear' [Colour figure can be viewed at wileyonlinelibrary.com]

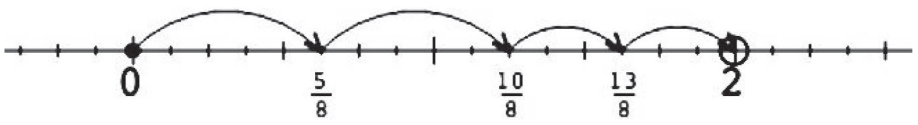

FIGURE 3 Adding eighths, like adding goats, gives more eighths. But there are surprises, too!

language influences their mathematics, we saw opportunities to work with even younger children.

To make the programming serve the mathematics (and have teachers recognise that), we needed to focus on key content, build a user interface with no distractions or overhead, and design the commands children would use in a way that conformed with how their mathematical ideas and language evolve (Sarama & Clements, 2020). We aimed to build mathematical microworlds that needed no more than a largely interactive 5 to 10 minutes on the rug before children moved to individual machines to solve mathematical puzzles independently using programming as their language. We were not building a teaching app but a *language* for expressing and exploring the mathematics particular to each microworld. Children could explore puzzles in any order; success was judged by them, not by the computer or teacher.

Our number-line microworld's variants span from early second grade (7-year-olds) addition and subtraction through the arithmetic (and notation) of fractions and decimals. Seven-year-olds see a number line with some 15 tick marks, only 0 labelled (not at the leftmost tick), a small number of programming blocks—for moves of −5, −3, +3 and +5, and for specifying a starting place START AT 0 (default 0 changeable by child)—and suggestions for explorations or puzzles (Figure 1).

The puzzles quickly evolve to offer new blocks, -500, -300, +300 and +500 and a 'zoomed out' number line with 'mathematically new' puzzles (Figure 2), taking advantage of children's linguistic sense that working with hundreds, though they have not yet studied it in class, is really *not* new to them.

By the next grade (8- to 9-year-olds), the same arithmetic ideas (still buoyed by linguistic sense that adding five of anything to another five should give ten of them) are applied to fractions, using commands like +⅝, −⅜ and so on (Figure 3).

Fractions, except when they represent integers, are expressed only as eighths (not fourths or halves or mixed numbers); as before, the environment allows (accidental or deliberate) excursions to the left of 0. This lets students build experience and knowledge from discoveries of their own, without the punchline or surprises being explained first. The surprises add salience and interest here, just as surprise *always* adds salience and interest.[8] That these puzzles feel so familiar as to be 'trivial' is the point; fractions are just numbers; $^3/_8 + ^3/_8$ gives $^6/_8$, not the canonically wrong $^6/_{16}$. Using $^6/_8$ rather than the reduced form $^3/_4$ helps remove the mystery. One puzzle says to 'Start at $^3/_4$ and then move to $^3/_8$'. Children may not anticipate the solution until they type into START AT 3/4 and click the block, which moves to the correct spot but labels it $^6/_8$. Equivalent values represented by different notations is interesting, and children will see plenty of examples, but first they need to realise that their intuitions about how numbers *should* behave applies to *all* numbers. That means fractions, too, which are often presented as somehow very different—'parts of' numbers, or 'pairs of' numbers, or parts of *things*—to which different rules apply.

In fourth grade (ages 9–10), children can zoom in between consecutive ticks initially marked in thousands, and see the line expand (and their tools change) to let them move among the newly revealed hundreds. They can then zoom in again between any two consecutive hundreds to see the consecutive tens between them (and new tools), and can zoom again to see consecutive 1 s, and *again* to see numbers with one decimal digit and so on. Each time, they get moves that are appropriate to the scale. Again, puzzles like those they saw in earlier years make clear that, eg, +0.003 behaves just the way +3, +300 and + ¾ did, and they build new experiences with scale. If 2 + 5 = 7, and if we understand (because we have played with it) exactly *where* 2.2 is, then 2.2 + 5 ought to be 7.2. This expectation has nothing to do with how we align numerals on paper. The alignment we teach for executing addition and subtraction algorithms is an artefact of the written notation and the algorithm (and is different for multiplication). That is helpful if we have complicated computations and need to keep track of them, but it is *not* the core mathematical idea and not a way to make that mathematical idea clear.

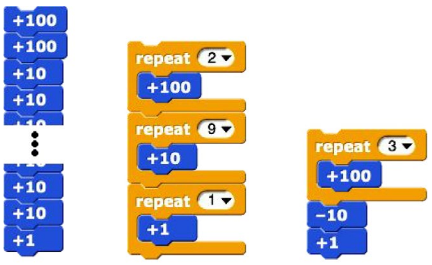

FIGURE 4 Three (of many) ways to create the number 291 using only ±100, ±10, ±1. The first resembles what a child creates with base 10 blocks, the second what schools teach about expanded notation and the third how a mental computation for adding 291 might best proceed, using rounding and adjustment [Colour figure can be viewed at wileyonlinelibrary.com]

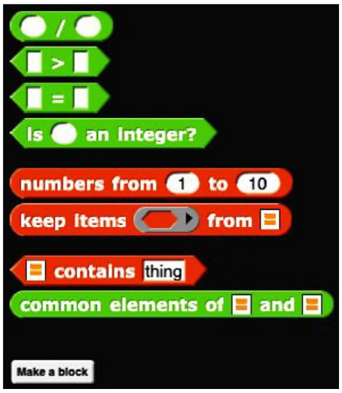

FIGURE 5 Some tools for exploring factors, multiples and common multiples [Colour figure can be viewed at wileyonlinelibrary.com]

Our early research showed that 7-year-olds can quickly learn to turn useful scripts—compositions of blocks (see Figure 4)—into new blocks that they find useful. For one example, the block that a girl built from the +3, +3, –5 script she had composed in the first few weeks of second grade (Goldenberg, 2019). Even so, we ultimately removed this opportunity for abstraction from *this* microworld for *that* year (though we used it in later microworlds at that same grade level), believing that, during the children's first contact with programming, it could be a distraction from the mathematics. This narrow commands-only experience in which even *order* does not matter much (order changes the pattern of arrows one sees but does not affect the solution to the mathematical puzzle) is *perfect* for children who have never programmed before and are intending to focus on their mathematics. But that means that despite all its mathematical versatility and value, the number-line microworld is not one that builds much programming skill even as the mathematics advances. To use programming as a language for mathematics, children's facility with the language must grow as their mathematics grows, so we must attend to that in other microworlds, which introduce blocks, control structures, predicates and functional programming in contexts *where their need and meaning become apparent*. The principle is always: *tools for solving problems, not tools for the sake of tools*. For example, in our second grade 1–10–100 microworld, children find it positively funny (which we believe means that it tickles and engages their logic) that they can create 291 in such very different ways as these (Figure 4).

In later grades, students need tools for experimenting with multiples and factors, for checking divisibility and for listing common elements of two sets (eg, of multiples of 3 and 4). Figure 5 shows one set of tools for a Multiples and Factors microworld, still being honed by experiments with 4th, 5th and 6th graders (ages 9–12). Except for recording results they have found, this involves entirely functional programming, advancing both mathematical and computational thinking.

At least in our initial trials, functional programming and use of lists—a style that is known in other contexts to be daunting to children of this age—appears to be quite manageable as long as the tasks are *semantically* clear to the children. Just as babies, toddlers and pre-schoolers learn their native language and vocabulary from use in context and not from definitions, 2nd through 5th graders learn computer vocabulary through use in contexts that seem to call for it. For example, one early puzzle in the Multiples and Factors microworld asks students to make a list of numbers from 0 to 5. Most 10-to-11-year-olds who have had prior experiences in our microworlds *anticipate*, without instruction, which block to use and what inputs to give it. For some other actions, like listing only multiples of 3, they do need an example to show that one can multiply a *list* by a number, though few of them seem

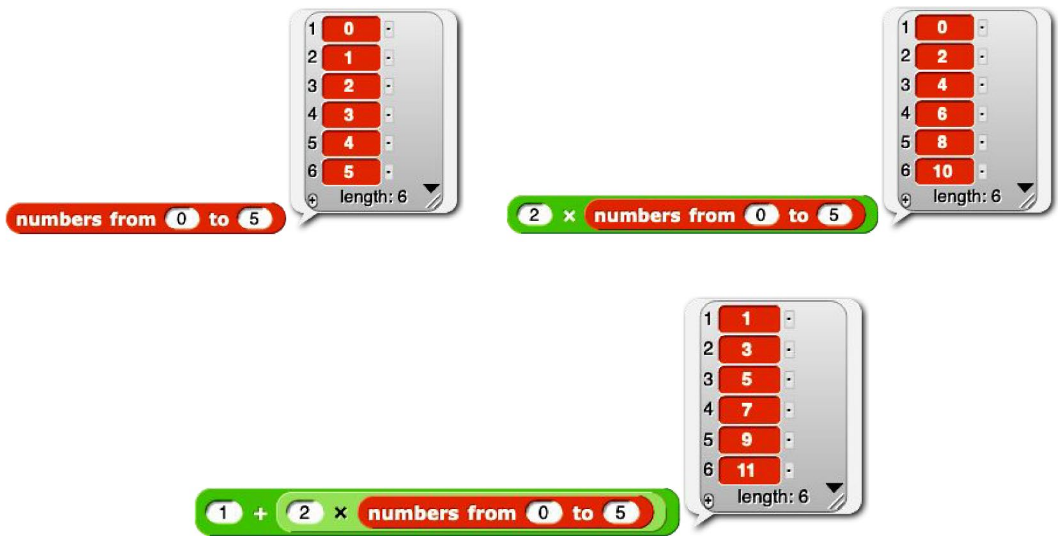

FIGURE 6 Making a list of consecutive integers, doubling those to get a list of consecutive even numbers and one solution for getting a list of consecutive odd numbers [Colour figure can be viewed at wileyonlinelibrary.com]

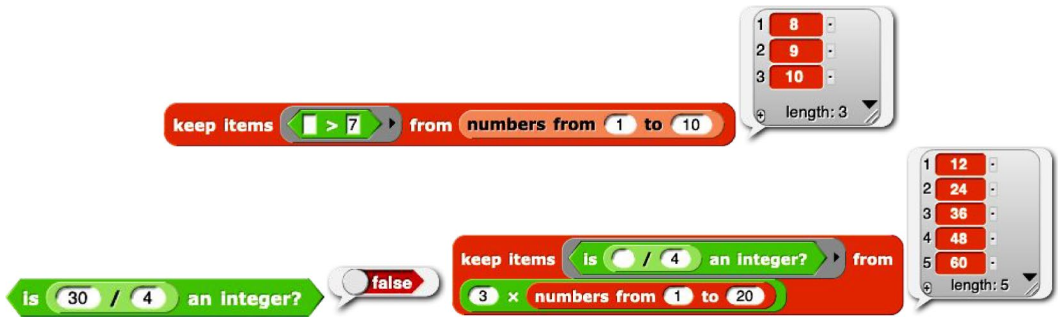
keep items (> 7) from numbers from 1 to 10
1 8
2 9
3 10
length: 3
is 30 / 4 an integer?
false
keep items (is / 4 an integer?) from 3 × numbers from 1 to 20
1 12
2 24
3 36
4 48
5 60
length: 5

FIGURE 7 The first script looks at a list of numbers and keeps only those that are greater than 7; the second shows how a predicate might be used to check for divisibility; the third shows one way of combing a list of multiples of 3 to keep only those that are divisible by 4 [Colour figure can be viewed at wileyonlinelibrary.com]

shocked that it is possible.[9] They can then do puzzles like 'make a list of *even* numbers from 0 to 10' and (typically far more challenging) 'make a list of *odd* numbers from 1 to 11' (see Figure 6). We think (but certainty would require more research) that the challenge is not in the programming. In fact, from watching 11-to-12-year-olds who are beginning algebra and just encountering $2n$ and $2n + 1$ as ways to represent even and odd integers, programming appears to support understanding rather than compete with it by making possible the kinds of experimenting that helps students with these new abstractions.

This raises what, on the surface, might seem like a crazy notion: done in some suitable way, teaching new programming ideas (functional programming, manipulating lists) at the same time as teaching new mathematical ideas (including building an essentially algebraic model for odd numbers) may be good, not bad. Programming's interactivity makes it different from the dead notations on paper that just sit there, correct or incorrect, and provide no feedback on their meaning or effect. The interactivity *helps* students build new mathematical ideas.

Will such a combination of new ideas all at once work broadly for 11-year-olds? We do not yet know. But first glimpses suggest ‘yes’ and we are already experimenting with the next step: checking numbers for certain properties and filtering lists for numbers with those properties. Figure 7 shows a sequence from tame to challenging. Here, there *is* a special syntactical surprise in the programming, the empty slot in the predicate > 5 and in the operator / 4 when they are used with **keep**. But that syntactical issue represents an important and new *mathematical* concept, a changeable member of an entire set of values to which a function is being applied. Any approximation of the written description we just used here is anathema to teaching—just too hard. But a few experiments, especially starting with the simplest of the expressions shown in Figure 7, give kids *experiences* from which they make their own sense. The most complex of those expressions takes a while because of all of the embedding, but it, too, is apparently tractable, and the shift *after* that to CMN seems then to be easier.

We are serious about the language part

At the outset with 7-year-olds, we talk about the computer as ‘speaking’ a language, Snap*!*, that the children will also learn to ‘speak’. The blocks are its ‘words’, and they can be *snap*ped together to make a ‘sentence’. We call that sentence a ‘script’ not to be fancy or technical but to be clear that we are talking about the computer's language, not our own. Kids understand right away, especially in classrooms where children *do* start with different languages. Children—all of the now several hundred we have worked with—are stunning in how much they remember from one three-lesson session to another that may be a month or two later. Teachers regularly express surprise at this retention, lamenting that what they more often see is children forgetting from 1 day to the next. Though we do not yet know for sure, our tentative theory about why this is so powerful is that the interaction and feedback is so very much like the way children normally learn from their actions (physical or linguistic) on the world. Despite the computer and an artificial language and the very ‘abstract’ mathematical focus, this conforms closely to the way children learn, complete with action, feedback, surprise and assimilation of new knowledge into old. It is, in that sense, ‘natural’, unlike instruction and practice on discrete skills, especially expressed in CMN. Perhaps that is why this engages them the way play does. To reiterate, we cannot yet explain the effect, but we can easily see the high engagement and clear understanding.

Language for thinking

Programming is also ideally suited to foster not only the mental practices typically associated with computer science and sometimes labelled, in aggregate, ‘computational thinking’ (CSTA, 2019; DESE, 2016; ISTE, 2019; K12CS, 2019; Wing, 2006), but also the mathematical habits of mind (Cuoco et al., 2012; Goldenberg et al., 2012; Mark et al., 2012) now codified in the US as the Mathematical Practice Standards (NGA/CCSSO, 2010), including these: (a) Programming provides a language with which students can ‘construct viable arguments and critique the reasoning of others’,[10] logically, providing a platform for constructing, experimenting with, testing and thinking about ideas; (b) it eases the process of ‘beginning with concrete examples and abstracting regularity’ or ‘look[ing] for and express[ing] regularity in repeated reasoning’); (c) used well, it can help students develop the disposition to ‘make sense of problems and persevere in solving them’ and ‘expecting mathematics to make sense, to feel coherent and not arbitrary, and to have understandable reasons behind facts and methods’; (d) it is a perfect medium for helping students learn to ‘attend to precision', as computers do what we tell them to do and when that is not what we expect, we can dissect our instructions to find the ambiguity or misstep and (e), of course, '[using] appropriate tools strategically', the tools being not just the computer and programming, but also tools like a number line used *strategically*: *not* just to get answers, but also to organise thought (cf. Clements & Sarama, 1997; Sarama & Clements, 2020). Finally, programming can support mathematical creativity—to many people, an oxymoron—posing new puzzles and inventing new ideas.

CONCLUSION

Our research was focused on a design challenge starting with the hypothesis that programming, if done 'right', could be a good language for children's study of mathematics. To test that hypothesis, we needed to learn how to design the 'right' programming microworlds.

We set many constraints on our work: focus squarely on mathematical content deemed critical for the grade; let the mathematics, not coding, drive the design, with coding and mathematics mutually supportive; require no more than 5 minutes introduction for children or their teachers about coding, no distraction from children's mathematics and no time taken from any other part of their day; generate high engagement by tapping curiosity and providing ease of exploration; let children's learning grow from their experiments to provide a foundation for later discussion, a pedagogy we refer to as 'experience before formality'. These were to be microworlds with mathematical integrity and high cognitive demand, ones that gave students experiences from which they could immediately construct their current-level mathematics and (then or later) also construct more advanced mathematics. And one core principle was to design for equity not by playing to pop-social images of 'boy' or 'girl' or 'white' or 'black' (or any other) cultural stereotype but by letting students personalise in their own ways and, critically, by offering challenge to all.

Unlike on-line tutoring apps or virtual manipulatives, we chose to have students interact via a language, with the thought that *articulating* their thinking was an important part of developing that thinking. Computer code has the precision of CMN, but is 'live' in that the computer's response shows exactly what it 'understood' the code to mean, without critique or commentary: the child evaluates the response. Understanding how children naturally use linguistic (and other 'non-mathematical' cognitive strategies) guided our work and the language elements—the programming blocks—we provide for children.

The focus of our research has been on understanding what is possible within these constraints, what design challenges the kids force us to meet and what puzzle types and structures we must provide. The data we collect helps us assure that we are at least meeting the self-imposed constraints: such things as time spent clearly on task (the math) and that the coding/tech aspects either contribute or, at least, do not distract by requiring special instruction. We checked for (and found) both easy entry and universal high engagement: puzzles were easy enough to be doable and hard enough to be fun. One research result that pleased us was that in these microworlds, students who had a history of low engagement and performance were as engaged as the others in their class, equally active achievers, and in some cases leaders. Our equity design aim was met.

But *of course* we get the kinds of results we just described! After all, our study is to teach us how to design and redesign in order to *make* those things happen. We observe results but cannot study them closely enough in this project to understand exactly which features by themselves or in combination cause those results. And, because this kind of experience in the classroom is so novel for the students, we cannot even rule out the Hawthorne effect for some of our observations (in particular, the startlingly universal engagement and the 'flattening' of the class by engaging those who are typically not engaged).

The results we reported earlier contain raise many questions for further research. We conjectured that because ‘interactive language can be very “concrete”’ and that because programming is also an abstract mental representation (a formal language) ‘it may be able support thinking in ways that [manipulatives] would not’. If that is true, it can be an important contribution to mathematics teaching and learning, but for now it remains a conjecture awaiting focused research.

From (limited) experiments with 11- and 12-year-olds representing even and odd integers with beginning algebraic ideas, we reported two conjectures: ‘programming appears to support understanding rather than compete with it by making possible the kinds of experimenting that helps students with these new abstractions’; and the ‘crazy notion [that]… done in some suitable way, teaching new programming ideas…at the same time as teaching new mathematical ideas…may be good, not bad’. Given the common counterargument that teaching two new things at the same time is not good, and given the widespread lament that students are ill-prepared for algebra, our observations suggest important follow-up research. Just how different is learning through active/interactive notation versus learning through notation on paper?

Our equity observation is extremely heartening, but we must carefully study who is served by the changes, not just rely on group observation. The work with programming appears equitable, in fact levelling, but we need certainty, not just appearance. With Covid-19 during the most recent year of our work, it became harder to tell if the levelling we had regularly seen in classrooms was still occurring, and we were frankly less certain. What factors are actually involved in the engagement and levelling that we know, with certainty, that we have seen?

Despite our strict focus on what is codified in the grade-level standards, as determined by current research to be the key grade-level mathematical content—a focus necessary for acceptance in school—we tested many of the environments at other grade levels. One interesting result followed from testing the second-grade number line, 1–10–100, and map microworlds with first graders slightly after the middle of their year. They were as engaged *and capable* as the second graders! Does that mean that the puzzles or content are too easy for second grade? It would seem not, because the second graders also puzzled and were engaged and evidenced learning. And some of the microworlds appear to have very much the same impact a grade later than we designed for. Why this huge range? Is the research on what is appropriate in what order and at what level just wrong? Probably not. That work was carefully done. What makes more sense is that learning with programming is just *different* from the learning that researchers have studied in paper/pencil-based classrooms. However, much the trajectories are influenced by cognitive factors or the structure of mathematics, they are also, in part, artefacts of the experiences children typically have in school. What we may be seeing is that the first graders can comfortably access ‘second-grade content’ when they can manipulate and get feedback and learn from their experiments, and that third graders still engage thoughtfully in ‘second-grade content’ because interaction lets them read more into the content than they drew from more static paper-and-pencil work. That is plausible but, of course, we do not *know*. Here, too, the apparent broad range of appeal—which may also account for the engagement of traditionally low performers that we noted earlier—could be a great support to differentiating learning without differentiating the learners.

We said that after a suitably designed programming experience ‘the shift…to CMN seems then to be easier’. This is another ‘seems’ that needs study. If that is genuinely the case, it would be a great contribution to mathematics teaching and learning.

We also noted the teachers' uniform surprise at the level of retention from one session to another, and its stark contrast to common expectations. This deserves to be treated as a full-blown result—it is large and, in our experience, universal—but it is not well understood. What causes this retention? Is it the novelty (again Hawthorne effect) of programming? If so, it may wear off if programming becomes standard practice. That one is challenging to research. Perhaps looking at the few schools where programming is standard practice. If the phenomenon holds up, we must also test our conjecture that retention is the result of creating an environment in school that more closely resembles learning outside of school: 'the interaction and feedback is ["natural," the way] children normally learn from their actions (physical or linguistic) on the world'.

And one obvious question is what this kind of experience does for students' mathematical learning as measured in conventional ways. That is not something that we could even plan to study at this stage in our work, though it is interesting to speculate that—as we noted above—learning through experimentation by programming is so different from paper-based learning that learning one way and testing in another may not reveal what we really want to know about students' growth in knowledge and reasoning.[11] The research must be done, probably with thoughtful assessment in both media.

So far, while not yet having large *n* verification of the effect, what we see is high promise worthy of use and further rigorous study. Our exploratory project, supported by NSF funding, is making the first steps in that research. The high level of engagement in genuine mathematical enquiry and action, all by itself, is clearly and obviously a win—no stats needed. Much remains to be learned including how mathematical and computational thinking will play out when assessments resemble a very different set of classroom experiences with which, as teachers report it, children are not as highly engaged.

ETHICS STATEMENT

All work with children aimed at refining and researching our work follows NSF guidelines, EDC guidelines and IRB, and school district guidelines and IRB, preserving de-identified data only. No formative or summative data are sharable. Links to the microworlds, as they are completed, will be available at https://thinkmath.edc.org by 15 October 2020.

ACKNOWLEDGEMENTS

We are greatly indebted to Douglas Clements for considerable support in the writing of this paper, including both encouragement and the significant work of checking and shoring up the background for our observations. Thank you, Doug! Funding for doing and reporting the work described in this paper was provided in part by the National Science Foundation, grants 1934161 and 1741792. Views expressed here are those of the authors and do not necessarily reflect the views of the Foundation.

CONFLICT OF INTEREST

The work involves no conflict of interest.

DATA AVAILABILITY STATEMENT

Links to several of the microworlds described in this paper are accessible on https://elementarymath.edc.org/ and more will be added as they become ready. Supporting materials can also be found there. The microworlds are freely available and licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License.

ORCID

E. Paul Goldenberg https://orcid.org/0000-0002-2077-1430

ENDNOTES

[1] Time erodes and simplifies written forms just as it does to languages themselves, but written forms are young by comparison, and conservative in some ways partly because reading and writing were long the private

property of the elite, not subject to weathering by common use. Even so, within the authors' lifetimes, *cooperate* has evolved from the earlier coöperate and co-operate, the words *data* and *they* are increasingly accepted as singular and *curricula* is becoming *curriculums*.

[2] To mathematicians, of course, CMN is 'live', too, because they can easily 'rerun' that code mentally. When fluent in a language, you can use it for talking to yourself. When you are just learning it, interaction with others gives needed feedback to tell you whether what you have said is what you meant and whether it follows the language's conventions.

[3] Not identical expressions, but understood as equivalent and, in children's early contexts, *intentionally* treated as interchangeable.

[4] Well, to be precise, neither is our visual processing of prose text, despite what we generally claim!

[5] Or, if printed, spelled out as we do here.

[6] Did you catch the error in this sentence?

[7] See, eg, https://www.codingforall.org/, https://www.csforall.org/ and http://www.bebraschallenge.org/.

[8] We often misunderstand memory to depend on repetition. Think, however, of a juicy bit of gossip, or a frightening accident you see. One experience and you remember! Regular occurrences of something that matters can also make it salient, but it is the salience, not the repetition, that makes us remember.

[9] Vocabulary, whether in natural or computer languages, is (more or less) arbitrary. Nothing says that the multiplication operation should take anything other than numbers as inputs. In fact, Snap*!*'s multiplication did *not* accept lists until version 6.0. But, for children, the (verbal) instruction to make a list of numbers on paper and then 'multiply those numbers by 2' is totally clear—they do not need a technical idea like 'map multiplication by 2 over the set of numbers you have written'—and so they may be pleased, perhaps even surprised, but not confused or bewildered at the ability of to behave this way.

[10] All quotations here are from NGA/CCSSO (2010, pp. 6–8), or Goldenberg et al. (2015, pp. 6 and 14).

[11] State-dependent or context-dependent memory (related but different phenomena) may influence what we see.

REFERENCES

Ball, D. L. (1992). Magical hopes: Manipulatives and the reform of math education. *American Educator*, *16*(2), 14; 16–18; 46–47.

Baroody, A. J. (2017). The use of concrete experiences in early childhood mathematics instruction. *Advances in Child Development and Behavior*, *53*, 43–94. https://doi.org/10.1016/bs.acdb.2017.03.001

Benton, L., Hoyles, C., Kalas, I., & Noss, R. (2016). Building mathematical knowledge with programming: Insights from the ScratchMaths project. In A. Sipitakiat & N. Tutiyaphuengprasert (Eds.), *Constructionism in action 2016. Conference proceedings*. https://drive.google.com/file/d/0BwnqbVelN16LbXY3NHJZaldQY3c/view

Benton, L., Hoyles, C., Kalas, I., & Noss, R. (2017). Bridging primary programming and mathematics: Some findings of design research in England. *Digital Experiences in Mathematics Education*, 1–24. https://doi.org/10.1007/s40751-017-0028-x

Bruner, J. S. (1985). Models of the learner. *Educational Researcher*, *14*(6), 5–8.

Clements, D. H., Fuson, K. C., & Sarama, J. (2017). The research-based balance in early childhood mathematics: A response to Common Core criticisms. *Early Childhood Research Quarterly*, *40*, 150–162. https://doi.org/10.1016/j.ecresq.2017.03.005

Clements, D. H., & Sarama, J. (1997). Research on Logo: A decade of progress. In C. D. Maddux & D. L. Johnson (Eds.), *Logo: A retrospective* (pp. 9–46). Haworth Press.

CSTA. (2019). *CS standards*. https://www.csteachers.org/

Cuoco, A., Goldenberg, E. P., & Mark, J. (2012). Organizing a curriculum around mathematical habits of mind. In C. R. Hirsch, B. Reys, & G. Lappan (Eds.), *Curriculum Issues in an Era of Common Core State Standards for Mathematics*. NCTM. (First appeared in *Mathematics Teacher*, 103(9), pp. 682–688, May 2010. Reston, VA: NCTM. 2010).

Cusi, A., Malara, N., & Navarra, G. (2011). Theoretical issues and educational strategies for encouraging teachers to promote a linguistic and metacognitive approach to early algebra. In J. Cai & E. Knuth (Ed.), *Early algebraization: A global dialogue* (pp. 483–510). Springer-Verlag Berlin Heidelberg.

DESE. (2016). Massachusetts Department of Elementary and Secondary Education. In *2016 Massachusetts Digital Literacy and Computer Science (DLCS) Curriculum Framework*. http://www.doe.mass.edu/frameworks/dlcs.pdf

Dewey, J. (1933). *How we think: A restatement of the relation of reflective thinking to the educative process*. D. C. Heath and Company.

Evans, R. L. (1973). *Jean Piaget: The man and his Ideas*. E. P. Dutton & Co.

Fletcher, P., & MacWhinney, B. (Eds.). (1995). *The handbook of child language*. Blackwell.

Gleitman, L., & Papafragou, A. (2013). Relations between language and thought. In D. Reisberg (Ed.), *The Oxford handbook of cognitive psychology*. Oxford University Press. https://doi.org/10.1093/oxfordhb/9780195376746.013.0032

Goldenberg, E. P. (1979). *Special technology for special children: Computers to serve communication and autonomy in the education of handicapped children*. University Park Press.

Goldenberg, E. P. (2019). Problem posing and creativity in elementary-school mathematics. *Constructivist Foundations*, *14*(3), 601–613. https://constructivist.info/14/3/319.goldenberg.pdf

Goldenberg, E. P., & Carter, C. J. (2020). How programming can serve young children as a language for expressing and exploring mathematics in their classes. In B. Tangney, J. Rowan Byrne, & C. Girvan (Eds.), *Constructionism 2020* (pp. 347–356). The University of Dublin Trinity College Dublin, Ireland, May 26–29, TARA, 620 pp. https://hdl.handle.net/2262/92768

Goldenberg, E. P., Mark, J., & Cuoco, A. (2012). An algebraic-habits-of-mind perspective on elementary school. In C. Hirsch, B. Reys, & G. Lappan (Eds.), *Curriculum Issues in an Era of Common Core State Standards for Mathematics*. NCTM. (First appeared in *Teaching Children Mathematics*, 16(9), pp. 548–556, May 2010. Reston, VA: NCTM. 2010).

Goldenberg, E. P., Mark, J., Kang, J., Fries, M., Carter, C., & Cordner, T. (2015). *Making Sense of Algebra: Developing Students' Mathematical Habits of Mind*. Heinemann.

Goldenberg, E. P., Miller, S., Carter, C., & Reed, K. (2017). Mathematical structure and error in kindergarten. *Young Children*, *72*(3), 38–44.

Guo, Y., Sun, S., Puranik, C., & Breit-Smith, A. (2018). Profiles of emergent writing skills among preschool children. *Child & Youth Care Forum*, *47*, 421–442. https://doi.org/10.1007/s10566-018-9438-1

Herbel-Eisenmann, B. A., Sinclair, N., Chval, K. B., Wanko, J. J., Clements, D. H., Civil, M., Pape, S. J., Stephan, M., & Wilkerson, T. L. (2016). Positioning mathematics education researchers to influence storylines. *Journal for Research in Mathematics Education*, *47*(2), 102–117. https://doi.org/10.5951/jresematheduc.47.2.0102

ISTE. (2019). *Computational thinking competencies*. https://www.iste.org/standards/computational-thinking

K12CS (K12 Computer Science). (2019). *K12 computer science framework*. https://k12cs.org/

Mark, J., Cuoco, A., Goldenberg, E. P., & Sword, S. (2012). Developing mathematical habits of mind. In C. Hirsch, B. Reys, & G. Lappan (Eds.), *Curriculum issues in an era of common core state standards for mathematics*. NCTM. (First appeared in *Mathematics Teaching in the Middle School*, 15(9), pp. 505–509, May 2010. Reston, VA: NCTM. 2010).

NGA/CCSSO. (2010). *Common core state standards for mathematics*. National Governors Association Center for Best Practices, Council of Chief State School Officers.

Noss, R., & Hoyles, C. (2018). *The ScratchMaths project is directed by Richard Noss and Celia Hoyles, with Ivan Kalaš, Laura Benton, Alison Clark Wilson, & Piers Saunders*. http://www.ucl.ac.uk/ioe/research/projects/scratchmaths

Papert, S. (1972). Teaching children to be mathematicians versus teaching about mathematics. *International Journal of Mathematical Education in Science and Technology*, *3*(3), 249–262.

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

Piaget, J. (1971). *Science of education and the psychology of the child*. Viking Press.

Sarama, J., & Clements, D. H. (2016). Physical and virtual manipulatives: What is "concrete"? In P. S. Moyer-Packenham (Ed.), *International perspectives on teaching and learning mathematics with virtual manipulatives* (Vol. *3*, pp. 71–93). Springer. https://doi.org/10.1007/978-3-319-32718-1_4

Sarama, J., & Clements, D. H. (2020). Technology in early childhood education. In O. N. Saracho (Ed.), *Handbook of research on the education of young children* (Vol. *4*, pp. 183–198). Routledge. https://doi.org/10.4324/9780429442827

Sendova, E. (2013). Assisting the art of discovery at school age: The Bulgarian experience. In *Talent development around the World* (pp. 39–98).

Sendova, E., & Sendov, B. (1994). Using computers in school to provide linguistic approaches to mathematics: A Bulgarian example. *Machine-Mediated Learning*, *4*(1), 27–65.

Sperry, D. E., Sperry, L. L., & Miller, P. J. (2019). Reexamining the verbal environments of children from different socioeconomic backgrounds. *Child Development*, *90*(4), 1303–1318. https://doi.org/10.1111/cdev.13072

Willingham, D. T. (2017). Do manipulatives help students learn? *American Educator*, *41*(3), 24–30.

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

How to cite this article: Goldenberg E. P., Carter C. J. (2021). Programming as a language for young children to express and explore mathematics in school. *British Journal of Educational Technology*, 52, 969–985. https://doi.org/10.1111/bjet.13080