Problem Posing and Creativity in Elementary-School Mathematics

E. Paul Goldenberg • Education Development Center (EDC), USA • pgoldenberg/at/edc.org

> Context • In 1972, Papert emphasized that "[t]he important difference between the work of a child in an elementary mathematics class and [...]a mathematician" is "not in the subject matter [...]but in the fact that the mathematician is creatively engaged [...]" Along with creative, Papert kept saying children should be engaged in ***projects*** rather than problems. A project is not just a large problem, but involves sustained, active engagement, like children's play. For Papert, in 1972, computer programming suggested a flexible construction medium, ideal for a research-lab/playground tuned to mathematics for children. In 1964, without computers, Sawyer also articulated research-playgrounds for children, rooted in conventional content, in which children would learn to act and think like mathematicians. **> Problem** • This target article addresses the issue of designing a formal curriculum that helps children develop the mathematical habits of mind of creative tinkering, puzzling through, and perseverance. I connect the two mathematicians/educators – Papert and Sawyer – tackling three questions: How do genuine puzzles differ from school problems? What is useful about children creating puzzles? How might puzzles, problem-posing and programming-centric playgrounds enhance mathematical learning? **> Method** • This analysis is based on forty years of curriculum analysis, comparison and construction, and on research with children. **> Results** • In physical playgrounds most children choose challenge. Papert's ideas tapped that try-something-new and puzzle-it-out-for-yourself spirit, the drive for challenge. Children can learn a lot in such an environment, but what (and how much) they learn is left to chance. Formal educational systems set standards and structures to ensure some common learning and some equity across students. For a curriculum to tap curiosity and the drive for challenge, it needs both the playful looseness that invites exploration and the structure that organizes content. **> Implications** • My aim is to provide support for mathematics teachers and curriculum designers to design or teach in accord with their constructivist thinking. **> Constructivist content** • This article enriches Papert's constructionism with curricular ideas from Sawyer and from the work that I and my colleagues have done.
> Key words • **Problem posing, puzzles, mathematics, algebra, computer programming.**

« 1 » Seymour Papert's early work and the origin of constructionism were largely outside of the school setting. The current school environment is even more rigidly constrained than it used to be. The question is, *Is there any hope for this kind of constructionist thinking and teaching in a school setting, not as a pull-out for well-resourced schools and with the best of their students, but as part of the regular program?* This target article shares some ideas that, to me, exhibit the essential elements of constructionism and could easily be core to even moderately conservative school practice.

« 2 » I, too, love playing with children outside the classroom. There is more freedom and it is easier. But we all know that if we are serious about touching many children's lives, we need a way to find them where they are. They are in school. Reaching them there is possible.

Children choose challenge

« 3 » Not all children and not all the time, but children do mostly choose challenge. Children are often pretty adventurous on the playground. Tiny ones climb the monkey bars higher than their parents are totally happy with. When climbing gets too easy, they hang upside down. Children walk on five-inch-wide retaining walls two to three feet above sidewalk level when they get a chance; they hop across the street on one foot; when bicycle riding feels easy, they try letting go of the handlebars. Even with games, they up the ante if the game feels too easy, changing rules fluidly to add extra challenge.

« 4 » For a toddler, there's enough challenge fitting the boat-shaped piece into the boat-shaped hole and the moon-shaped piece into the moon-shaped hole, but when that's no longer a challenge, children seek more. Kindergarteners like fitting together the two-dozen jigsaw puzzle pieces of a large picture of a dinosaur. And when that gets too easy, some try putting the pieces together face down, some try jigsaw puzzles with smaller and more numerous pieces, and some just move on to totally different activities.

« 5 » Children also put effort into figuring out how things work. Laura Schulz and Elizabeth Bonawitz (2007) showed preschoolers a box with two levers and two different toys that popped up when the levers were pressed. One group of children were shown that each lever caused one toy to pop up. The other group saw only that when both levers were pressed simultaneously, both toys popped up. The first group's information was complete and unambiguous, with nothing left to figure out. The second group's information was incomplete: either lever might have controlled both toys, with the other doing the same, or nothing, or raising just one toy if pressed by itself; or the two levers might be totally independent, one for each toy. When the children were then given the toy to play with (or ignore) on their own, children in that second group played longer, spontaneously exploring to puzzle out the cause-and-effect relationship. It is tempting to relate the first group's experience to the situation children often experience in school mathematics, where common pedagogy (at least in the US and UK) shows exactly how each thing is done, leaving no evidence that there *is* anything to figure out, and taking little advantage of children's built-in curiosity.

« 6 » Children also love riddles – challenges to logic, interpretation or perception. And just as they spontaneously add challenge to their playground activities or jigsaw puzzles, they will add to their repertoire of riddles and jokes by making up their own, sometimes creations that they, at their age, find funny (illogical) because the challenge "works" for them, and that we adults find simply ludicrous (illogical) because the challenge no longer works.

« 7 » The common feature is the challenge. When it is not there, children are bored. When they are bored, they invent challenge. In school, that mischievous inventiveness can be to the dismay of their teachers, whose response may dismay the children, but even that will not stop the children's drive for the challenge.

Why puzzles?

« 8 » Kittens stalk and pounce to make their hunting skills sharp, and they scratch to keep their claws sharp. That is because sharp claws and hunting skills are among the particular adaptations that make their species successful. *Our* species' special adaptation is not sharp claws and pouncing but a mind that lets us adapt to nearly any environment, which is how we wound up populating city and farm, blazing heat and frigid cold, arid desert and tropical jungle. Keeping our minds sharp is what makes our species successful. Evolution built a mechanism to nudge creatures to repeat those actions that, for their species, are most useful. Humans experience it as pleasure. We *like* the feeling we get when we stretch our minds, so we seek it out.

« 9 » Having evolved to adapt to such varied environments means we start with less "built-in knowledge" about which features will matter most for survival. We must figure that out. That has implications for learning. As children, we watch social behavior (Whom should we copy, stay near, avoid?), animals (Are they food, playmates or danger?), artifacts (How do they work?), math lessons (Who knows? Maybe they are important) and everything else. Little children listen closely to the words others use, and repeat them, whether or not they relate to the current activity. In *our* species, it is adaptive for the young to be distractible and not to focus too narrowly; some "attention deficit" is natural, and a built-in *asset* for a child.

« 10 » For adults who must focus to "earn a living," whether that is by blow-darting the rabbit (while avoiding the tiger) or by generating research papers or teaching children, allowing their attention to wander is not as adaptive. But adults still have to keep their minds sharp. Adults argue about ideas – politics, religion, what to wear, business plans, the lives of others, predicting which team will win or what their best strategy is – even when the practical value of the argument is near zero. It is a mental workout.[1] Mental challenge is not just for academics; *all* people whose minds are not already fully occupied finding food or avoiding danger seek ways to keep their mind busy. Mental challenges for adults are sold not just in academic bookstores but also in supermarkets; puzzles appear in newspapers and in airplane magazines. Boredom is painful; enforced boredom is torture.

« 11 » Several things distinguish purely recreational "puzzles" from standard school problems. The most obvious is that they are optional. But tasks designed for educational purposes – non-optional and non-recreational – can also be designed in ways that tap the same drive that moves people to take on optional puzzles, a drive Marcel Danesi (2002) refers to as "the puzzle instinct." Tasks that generate surprise can stimulate curiosity and the eagerness to satisfy that curiosity by exploring more. Puzzles require puzzling: searching, figuring out what to do, and a bit of time. A crossword puzzle provides some hundred clues. Even if the individual clues are not obscure and "tricky," it is not immediately obvious which clue to use first. One searches for a place to start, tries an experiment, confirms or rejects the word, and then moves on. The *content* is not mathematical, but the way of thinking – that search for an entry point and for data that supports or weakens a conjecture – is very close to one element of mathematical practice we want students to develop. (And, of course, tasks that do have mathematical content can also offer that experience of genuine puzzling and surprise.) Solving a puzzle is different from working an exercise: the process is not rote or algorithmic, not just the application of some technique one just learned.

Puzzles and surprise in mathematics learning

« 12 » In 1964, Walter Sawyer (2003) seeded the ideas for a wonderful textbook series for primary-school mathematics (Wirtz et al. 1964) and for our own curriculum materials (see, e.g., Goldenberg & Shteingold 2007a, 2007b). He took a very algebraic approach to teaching elementary arithmetic, with a major emphasis on play and surprise. On the surface, the content was exactly what one expects for the grade level but with a twist that included research, puzzles for children to figure out, all foreshadowing the algebra that children would learn later.

« 13 » For example, as a way to give seven-year-olds practice with addition and subtraction they start with a piece of mathematical research. A child is asked to suggest some addition equation like $4+2=6$ or any other, and the teacher would write it on the board. Another child is asked to suggest a new equation, e.g., $1+2=3$, which the teacher carefully lines up directly underneath the first. Then the teacher has the children add vertically, displaying the results as in Figure 1 (left).

1| Children also practice intellectual argument, debating rules of games, veracity of claims, or meanings of words and ideas. Those arguments share many characteristics with play even when, to our adult eyes, they seem to be getting in the way of the play.

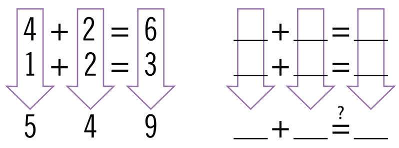


Figure 1 • Left: "Adding" two equations. Right: A blank for children to experiment with.

« 14 » Do these three new numbers make a mathematically correct addition equation? The teacher completes the bottom row of numbers to read $5+4\overset{?}{=}9$. Surprise!

« 15 » Will this always happen, or did they just get lucky? Children are given the challenge of finding a pair of addition sentences that do not work (Figure 1, right). Seven-year-olds are sure they can find some and set off busily, getting lots of practice.

« 16 » Of course, they *will* find some (they think) and report them excitedly, but the preponderance of cases that do work will get even seven-year-olds to doubt the counterexamples and check to see if they have made a mistake. This research is hardly a project in Papert's sense – sustained, active engagement, like children's play, in a programmable research-lab/playground. This tiny research project may not even last a full classroom period, but it does generate curiosity, the creative engagement that Papert referred to as the experience of the mathematician. Note also that it is not just a school exercise "gamified" but a mathematical result that is surprising and generates curiosity.

« 17 » Their research convinces them of a result, but if we do not leave it as magic and instead help expose the logic inside the puzzle, children get even more excited. They have a tool they can and do use, first to figure out for themselves why the puzzle works and then to invent new puzzles for themselves and their friends! Exposing the logic involves reminding children of reasoning they developed in Kindergarten and first grade. Given a collection of buttons differing by two attributes, color and size, kindergarten children naturally sort, though sometimes their sorting is idiosyncratic – two large buttons and a small one, for example, to make a "family." They learn to respond to "show me a small button" and "how many small buttons do you have?" And they can learn to respond to "show me a large grey button" and "how many small blue buttons do you have?" After sorting by a single attribute (Figure 2, left), they can learn to sort by two attributes (Figure 2, right).

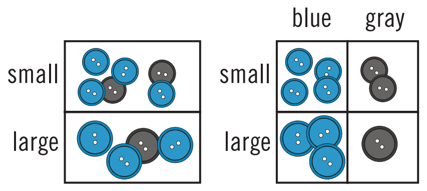


Figure 2 • Sorting buttons in kindergarten.

« 18 » *Now*, when we ask how many small buttons and how many large, we are summarizing the rows, and we can write that summary (Figure 3, left). And we can do the same for the columns, summarizing the number of buttons by color (Figure 3, right).

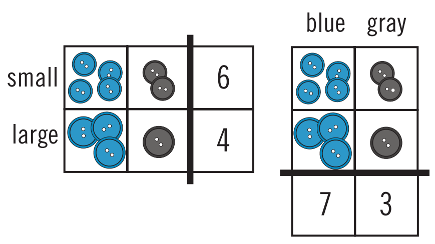


Figure 3 • Recording data from the sorting.

« 19 » Once children grasp and can use cardinality, it is clear that the number of blue and grey must be the same as the number of large and small – either way, it is all the buttons. Second-graade students comfortably replace buttons with numbers (Figure 4) and then use that structure as part of their reasoning.

« 20 » Reading across (Figure 4, right), children see $4+2=6$ and $3+1=4$; adding down the columns, they get 7, 3, and 10, which *must* make a correct addition statement.

« 21 » Subtracting down the columns is not always possible for seven-year-olds – depending on the situation, it might require negative numbers, and the meaning changes, too (it does not yet make sense to subtract the number of large buttons from the number of small ones) – but with numbers that they can subtract (as is the case in Figure 4, right), the arithmetic still works and produces a mathematically correct addition statement. Subtracting to see how many more small buttons than large, we get $1+1=2$. And that exact same logic will be essential in algebra a few years later! (Figure 5)

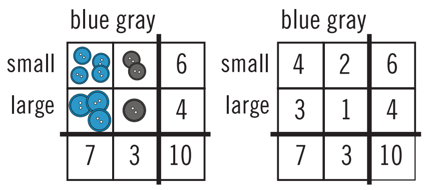


Figure 4 • Buttons (left) replaced by the number of buttons (right).

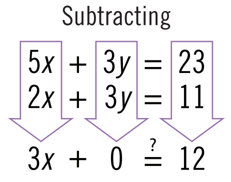
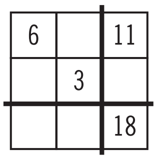


Figure 5 • Algebra subtracting equations.

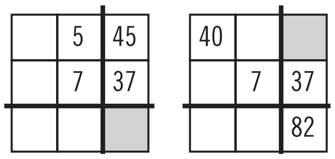
« 22 » The format is not just a school artifact; it is the structure of any spreadsheet that subtotals the columns and rows and has a grand total. Robert Wirtz et al. (1964) used this format as a puzzle. (Which cell in Figure 6 might you fill in first?)



6		11
	3	
		18

Figure 6 • A puzzle based on a spreadsheet with subtotals and grand total.

« 23 » They also used it as a route into multi-digit addition and subtraction (Figure 7).



	5	45
	7	37
		▨

40		▨
	7	37
		82

Figure 7 • The same arithmetic presented (left) as an addition puzzle 45+37, with the grey square as the sum, and (right) as a subtraction puzzle 82−37, with the grey square as the difference.

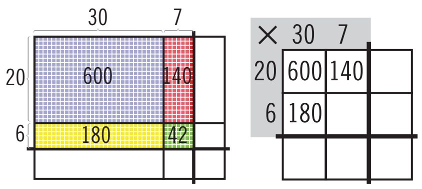
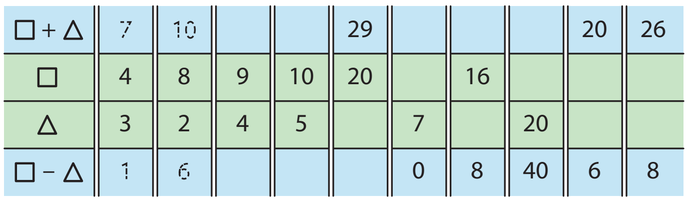


Figure 8 • An array model of multiplication. Left: proportional to scale. Right: abstracted.



□ + △	7	10			29				20	26
□	4	8	9	10	20		16			
△	3	2	4	5		7		20		
□ − △	1	6				0	8	40	6	8

Figure 9 • A practice exercise for 2nd grade, foreshadowing systems of equations (after Wirtz et al. 1964).

« 24 » For nine- or ten-year-olds, this structure also models the multiplication algorithm. Instead of color and size labels, we label the width and height of the columns and rows, and imagine cells filled with unit squares instead of buttons. How many squares are in the four regions? In the most concrete image, everything is to scale (Figure 8, left).

« 25 » With a smaller array, say 3 × 4, we can see why multiplication gives the answer and we can count to check. But with numbers like 37 × 26, we certainly do not want to count! Instead, we use an abstraction (Figure 8, right), ignoring scale, but maintaining a sense of the logic of multi-digit multiplication, not a set of memorized steps that often wind up feeling arbitrary. Of course, the steps involved in this logical model map perfectly onto the abbreviated notation often taught in school, and fully explain that notation. Moreover, it is worth delaying the abbreviated notation until children are so secure in the logic of the array model that they can easily extend it to three-digit multiplication, because exactly this method – four separate multiplications and only then a possible summing up – will be required when the students study algebra.

« 26 » Sawyer suggested other ways that even very young elementary content like addition and subtraction of small numbers could be learned or practiced in a puzzle-like context that both builds curiosity and foreshadows later ideas and methods. Figure 9, for example, shows what, in standard worksheets, might be presented as 16 unrelated addition/subtraction practice exercises for seven-year-olds, but structured here in a way that adds a bit of intellectual challenge – how-do-I-do-this? – and foreshadows systems of equations that the children will meet several years later.

« 27 » Again, it is not a "project" in Papert's sense, and not "creative" in the most familiarly used sense of that word, but especially the last two columns pull for children to be *mathematically* creative.

« 28 » One of the most powerful introductions to algebra that I have seen is what Wirtz et al. (1964) called Think-of-a-Number tricks. For example: "Think of a number. (Yes, you! Please think of a number.) Add 3. Double the result. Subtract 4. Cut that result in half. Subtract your original number. Aha! I can read your mind! You got 1 at the end!"

« 29 » For nine- or ten-year-olds, this is wonderful magic. They want to do it over and over, but also want to know how it works. I say that I picture the secret number as that many marbles (or grapes or whatever), tucked in a bag or bucket where we cannot see them – only the secret keeper knows the number inside. When I give the instruction "add 3," I know about those marbles, so I draw them outside the bag. I ask the children what the next instruction is (they almost always remember) and what the picture should be like (they almost always say "two bags and six marbles"). Then I continue, each time asking the children to describe the next picture. At the end, "subtract your original number" gets rid of the bag. So, the number of marbles in it does not matter! There is one marble left, and we can *see* it!

« 30 » Even after the usual huge smile and the cry "I get it!!," seeing it once is not enough. The understanding evaporates until children see the generality, not just the way this particular trick worked. To create that abstraction for themselves, children need research time: practice drawing pictures to match instructions, applying instructions to specific numbers, and variations on the trick from which to generalize and learn to invent their own tricks.

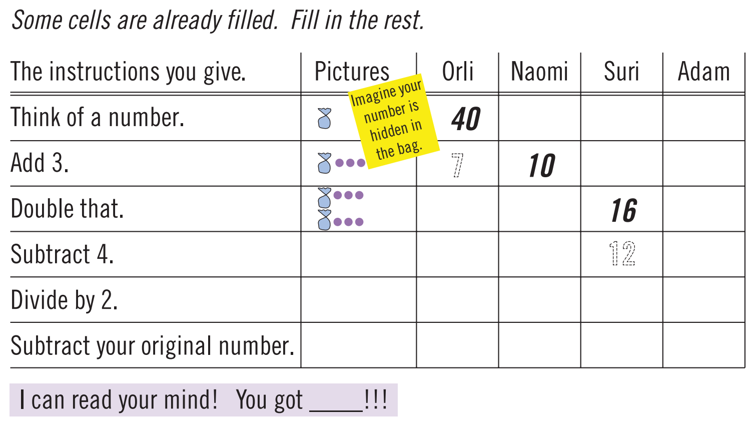
« 31 » They also need chances to study the trick inside out and backwards (Figure 10), starting, for example, with the 16 that Suri had in mind after the instruction "double that" and figuring out what secret number she must have started with. To do that, a child might note that the picture corresponding to Suri's 16 *shows* six of those marbles, so ten marbles must be hidden in the two bags. Suri's secret number – the marbles in *one* bag – must have been 5.

« 32 » I have recently been introducing a new crop of eight- and nine-year-olds to algebra this way and told them that they would soon know how to invent new tricks of their own. After two days of playing with the puzzle, Lucy said "I really get it, but I still don't know how to make up my own." So, we played. I said "OK, I've thought of a number" and I drew . "Just make up one instruction, anything you like, and I'll draw the next picture." She said "add 5?" I said "OK," drew ●●●●●, and asked "What next?" She said "double that?," still with the question in her voice. I said "whatever you'd like me to do… Is that what you want me to do?" She nodded and I said "you draw the picture." She drew two buckets and 10 dots. She then told me to subtract 2 (no question in her voice, and she drew the picture), then subtract 7 (she drew the picture). That change in tone – no question in her voice – was because she now understood something new, not just about the mathematics of this trick but about mathematics, itself. She could make up a rule, *any* rule, and it was then up to her to figure out its implications. That is so much like watching a child program, see the effect, decide whether that effect is desired or not, and then decide what to do next.

« 33 » I asked, "OK, what can you do in order to know my number?" Long pause. Then Lucy commanded "subtract your original number" (and drew the picture). After another pause, she said "Oh!! Subtract your original number *again*!" Her smug smile showed clearly that she knew what she had done but I wanted to check, so I prompted her to "read my mind." Instantly, but with excitement and what also sounded like surprise in her voice, she said "Oh! One! You got one!" as if understanding the trick for the first time all over again. The joy of "getting it" is far more magical than any grade, praise or prize could be.

« 34 » These are 5- to 15-minute events. By the end of a week of them, instead of drawing the pictures that the children describe, we write the words with which they describe the pictures. "Two bags and six marbles" is a lot to write, so we abbreviate it: $2b+6$. No discussion of variables; no explaining about letters standing for numbers; $2b+6$ is terse, but represents the language the children themselves used, and they fully understand it. For now, that's enough. Later, when they formalize algebra, the bag or bucket image is useful to return to: a variable is a container for a value.

« 35 » *Containing* a value (or being a *pointer* to it) is the programmer's image; *representing* a value is the mathematician's image. The underlying idea common to both images is that a value can be referred to by a name and that this abstraction is useful. In practice, nearly all children love the think-of-a-number tricks, so they become a natural, appealing and compelling way to acquire that value-naming idea. Part of the power of the "trick" is that it is faithful to the mathematics, even though it is limited.[2] But part of its power, I am sure, is what Schulz and Bonawitz (2007) saw: children play longer and more curiously when there's something they do not understand *and they believe that they can figure it out*.[3]


Some cells are already filled. Fill in the rest.

The instructions you give.	Pictures	Orli	Naomi	Suri	Adam
Think of a number.	Imagine your number is hidden in the bag.	40			
Add 3.		7	10		
Double that.				16	
Subtract 4.				12	
Divide by 2.					
Subtract your original number.					

I can read your mind! You got ____!!!

Figure 10 • Using bags and marbles to introduce 3rd graders to algebraic notation and solving equations. (Idea from Wirtz et al. 1964, reworked for 3rd grade based on Mark et al. 2014 and Goldenberg et al. 2015).

« 36 » This was not a classroom assignment. The children did not have to do this and would not be tested on it. But they put effort and attention into the think-of-a-number trick because they *want* to know how it works. The intensity of Lucy's interest, even readily admitting what she could not yet do and asking for help doing it, was because there was a genuine mystery left to solve – one that she saw as *hard* – but she was so tantalizingly close that she was convinced she could reach that goal.

Why have students invent puzzles?

« 37 » Four reasons come immediately to mind; perhaps there are more.

« 38 » First, the construction of a workable puzzle is a creative act, making the student a creator and not just a consumer of mathematics. We who call ourselves constructionists easily accept making as a good thing, but it is useful to say why. What you make is yours; creating gives ownership. Mathematics is often perceived – except by mathematicians – as the antithesis of creativity, a subject in which rules rule and we obey. It is very possible to learn mathematical content that way, and some people like that order and simplicity. But mathematical thinking cannot work that way because genuinely new problems could then never be solved. For new problems, one *must* create new ideas and approaches. Young students' mathematical creativity cannot be at the leading edge of mathematics, but it can be at *their* leading edge. Puzzles are not the only opportunities for students to be creative in mathematics but they are good ones, especially for younger students.

« 39 » Second, constructing a good sharable puzzle is a balancing act – easy enough to be solvable and hard enough to be fun. To be solvable, a puzzle must also be well specified – enough clues to derive a unique solution (or a limited class of solutions) – without having so many clues that only the arithmetic is left. Determining when one has given enough clues to derive a solution is quite a challenge.[4]

2| This imagery does not represent "divide by 2" well unless the numbers of bags and marbles are both even. The imagery is adaptable to "negative marbles," but frankly awkward. So, we need to be clear that the imagery is not the goal, not a "new method" for algebra. But it is an extremely effective entry to algebra.

3| This qualification is important. *Nobody* – no corporation, no person – puts time/money/effort into an endeavor that they believe has no chance of success. Students who have been convinced they are "no good at math" often do not put effort into study that *we* believe would make them better. But *they* do not share that belief, so from *their* perspective, it is wiser to aim their efforts in a direction that seems more likely to pay off. That is an adaptive, economical choice. That is why it is so important to show (not tell) them that they *are* capable by hooking their interest on something they perceive as hard but attainable.

4| This is especially the case when creating a good MysteryGrid puzzle or Who Am I puzzle, not described here, but part of the SolveMe suite of puzzles mentioned in §42.

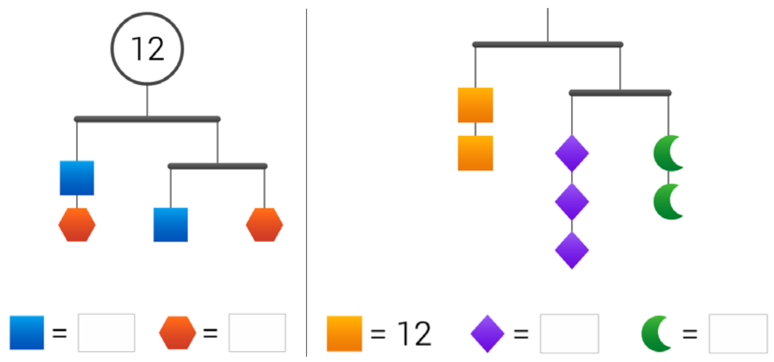


Figure 11 • Two relatively simple mobile puzzles.

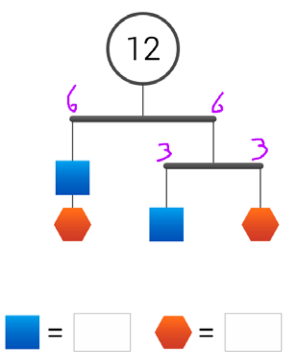


Figure 12 • Annotating a mobile puzzle.

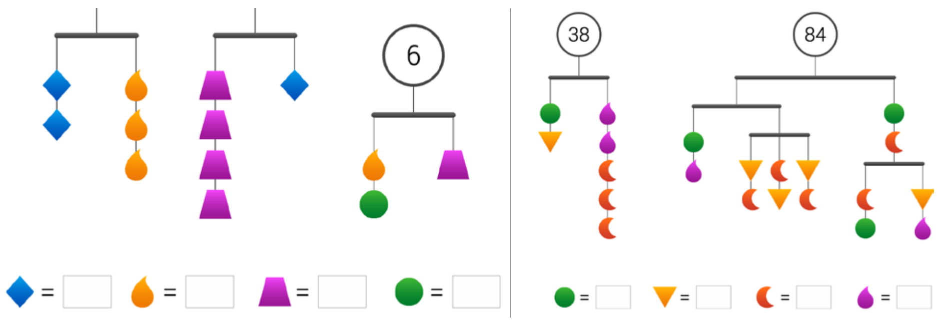


Figure 13 • Two mobile puzzles at a more advanced level.

« 40 » That challenge, and also the act of being a creator, may be part of why construction of a sharable puzzle appeals to children, but the appeal is yet a third reason to have children create.

« 41 » And fourth, construction of a sharable object helps reveal the child's thinking to both the child and teacher, supporting refinement of that thinking, and discussion and analysis.

« 42 » SolveMe.edc.org is a puzzle world with three kinds of puzzles aimed at developing algebraic reasoning. Each puzzle type also lets students create their own puzzles and share them on line. The Mobiles app collection begins with relatively elementary puzzles like the ones in Figure 11, and even simpler ones for complete beginners.

« 43 » The mobile's total weight might be given (Figure 11, left) and players must figure out how much the blue red objects must weigh in order for this mobile to balance. Or (Figure 11, right), no total weight might be given, but the weight of one of the hanging objects might be specified. Again, the player must puzzle out the weights of the other objects.

« 44 » Players often just work these out in their heads, but the app offers other options: they can scrawl annotations on the screen (Figure 12).

« 45 » They can also create equations by dragging off a copy of a horizontal beam ■ = ⬢, or the entire mobile 12 = 2⬢ + 2■, and substitute these into other equations (or the mobile) to derive new information, like 12 = 4⬢. The app also lets them factor 2 out of equations like 12 = 2⬢ + 2■ to derive new equations 6 = ⬢ + ■ and to drag a common element out of both sides of an equation like 4● = 2▼ + ● to get 3● = 2▼. Mara Otten et al. (2017) describe how eleven-year-olds used explicitly algebraic correct reasoning in the context of informal notation and manipulations of a physical hanging mobile.

« 46 » The mobile puzzles are essentially systems of equations. Some students are intrigued that they can get those equations and see what those equations mean. In class, that is an advantage, but informally, even the students who like that they can get equations mostly do not work with the equations, instead inventing informal methods equivalent to the formal manipulations that algebra classes teach and name. They also see, early on, that the "weights" can be fractional and even negative.

« 47 » Some of the puzzles are quite challenging, like the ones in Figure 13; without being required to, students persevere because they are sure they can solve the puzzles if they keep at it.

« 48 » As I had said, we felt it important to provide a tool with which students could create their own puzzles and even share them with friends or with the entire SolveMe community. The sheer variety of users' contributions is fascinating. Some are genuine puzzles, like those shown above. Others seem to be intended more as works of art, like the ones in Figure 14.

« 49 » Tracy Manousaridis (personal communication 2018) regularly encourages students in grades 2 and 3 to create their own puzzles as posters after solving some online. Part of her goal is, of course, the ownership that comes from building a puzzle. But it is also clear that the task naturally leads children to work at the frontier of their ability, partly because they take special pride in pushing (and displaying) what they can do.

« 50 » The nine-year-old who created the puzzles shown in Figure 15 was clearly proud of the arithmetic she did but especially proud of having created a puzzle that required such fancy arithmetic. The puzzle, not just the artwork on the poster, is a highly personal and creative act. This child is what Papert (1972) described as "the mathematician […] creatively engaged."

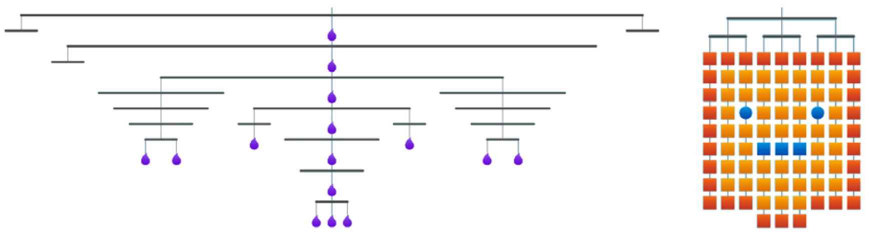


Figure 14 • Two mobile "puzzles" invented by users, apparently intended only as art.

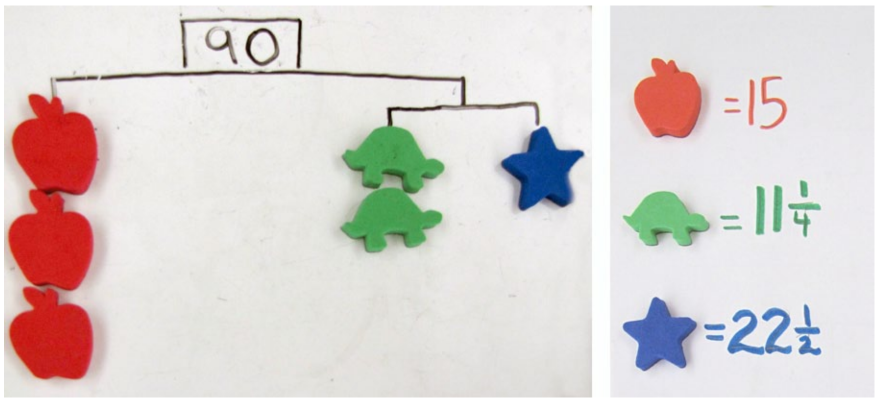


Figure 15 • Photographs of a mobile puzzle invented by a nine-year-old to challenge her classmates to use fractions.

Programming as a language for learning mathematics

« 51 » The examples and contexts described above have been very far from the programming-centric proposal that Papert made in 1972. But they are well in line with the mathematical creativity, exploration, and research projects that he regarded as *doing* mathematics rather than learning about it – solving genuine puzzles and creating one's own versus learning mathematical facts and solving exercises created by others. While nobody would claim that programming is the only (or even always best) venue for creative expression and exploration in mathematics, I and others believe it can be an enormous help if it can become a language for and a natural part of learning mathematics. For it to be "a natural part," it must develop along with the mathematics, growing over time as the mathematics does, and used in ways that support the mathematics and do not compete with it. That is, it must not be, nor even seem to be, a separate venture – fun stuff but disconnected. It must not be overhead or distraction. If that can be achieved, then the flexibility and expressive ability of programming can give it a central role in children's mathematical learning and creativity.[5]

« 52 » Richard Noss and Celia Hoyles focused especially on that expressive ability:

“Maths is difficult in part because of the language in which it is expressed. Can we find a different language – and set of ideas and approaches – that is more open, more accessible and more learnable. And can we find it without sacrificing what makes mathematics work? Our tentative answer is 'yes' – the language of programming might, if we design it right, be just such a language.”[6]

« 53 » Mathematics needs three languages. Two are already used universally in school: natural language for semantics (context, explanation, and some of the logic) and conventional arithmetic (algebraic) notation. Both are necessary but, if used inappropriately, both can also get in the way. For young children, mathematical notation is best used as a clean and concise way to record ideas that the children already understand well, not as the entry point to new ideas, as appears to be nearly universal practice.[7] Here is why. Recall that the third grad-

5| It is important to emphasize that programming, here, is not promoted as part of the current enthusiasm about "coding for all," which is often associated with claims about viability for the work force. The ability to talk to machines the way programmers (currently) do may turn out to have some job value but smacks of a dubious promise. It feels similar to the claim that one cannot survive without mathematics in the 21st century, a mantra that everyone is happy to repeat even while knowing so many people who unashamedly say they are "not good at math" and yet are surviving quite well.

6| http://www.ucl.ac.uk/ioe/research/projects/scratchmaths

7| This parallels teachers' understanding that writing is a *record* of language, and that understanding the meaning of the language comes first.

ers knew intuitively that doubling ●●●●● produced ●●●●● ●●●●●. Moreover, six-year-olds, when asked verbally (not in writing) what five eighths plus five eighths might be, are happy to respond "ten ayfs"[8] and then, perhaps, even ask "what's an ayf?" (Goldenberg & Carter 2018). They never answer ten sixteenths. The distributive property is built in to our logic early. But when the term "distributive property" is introduced in third grade (in the US, that is a commonly mandated content standard), it is often taught with a written string like $8\times 7 = 8\times(5+2) = (8\times 5) + (8\times 2) = 40+16 = 56$ that is opaque and daunting to a beginner. It is likely that, despite your own mathematical literacy, knowledge, and adult cognition, even you zipped past the string of symbols without reading closely enough to see if it was typed correctly. Processing such a string of symbols takes focus and effort, so it cannot be the optimal way to *introduce* the distributive property to an eight-year-old. Too much cognitive space is taken up just decoding the long string; not enough is left for thinking about the idea.

« 54 » But part of learning to reason mathematically involves focusing on the steps one takes to solve a problem. Neither natural language nor mathematical notation is particularly good at expressing process or algorithm. That is what a good programming language can provide. Also, unlike a string of symbols or words that sits on paper, correct or incorrect, and gives no feedback without the reader (re)reading and (re) processing it mentally (or relying on outside authority to validate it) a programming language is a notation that can be *run* and will give direct and clear feedback. Papert's idea was that programming offered new contexts and opportunities for engaging in mathematics. This notion of programming as a *language* for learning and expressing mathematics is a bit different (see, e.g., Sendova & Sendov 1994; Sendova 2013) and is explicitly stated as a rationale behind ScratchMaths (see Footnote 6).

« 55 » ScratchMaths is one beautiful example of infusing programming directly into grade-level-required mathematics for nine- to eleven-year-olds. At EDC, we are extending that range, building programming into elementary-school mathematics for children aged 6 through 11. This new work – currently focused on second grade (seven- to eight-year-olds) – builds on *Think Math* (Goldenberg & Shteingold 2007a, 2007b), inspired by the brilliant, playful, puzzle-centric ideas of Sawyer (2003) and Wirtz et al. (1964), described earlier. It is driven by state-required mathematical content and practice, not by presumed computational thinking (CT) and computer science (CS) goals, building programming content and skills as needed to serve mathematical purposes. But, of course, to serve the ultimate goal of giving children a language for their mathematics, it must, over time, also develop programming, not be limited to a few basic commands, not be an app for teaching math. Though the necessary constraints presented by the formal requirements of state-wide schooling narrow the range of programming projects we can choose, the puzzle/surprise/research principle can survive quite well, even when constrained by conventional content.

« 56 » Initial programming experiences for young children can be quite open – directing the actions of a robot, or even just code-streams of interesting effects – but if the explicit intent is to give seven-year-olds a language that lets them experiment with and express the mathematics they are learning, the first coding experiences must be simple enough not to be distraction or overhead, must be directly connected with the mathematics they are learning, and must be full of room for puzzling and exploring. To keep the intellectual focus on mathematics – not the mechanics of typing or the placement of semi-colons – our team chose the blocks-based language Snap*!*, motivated by and visually similar to Scratch, but with capabilities and constraints optimized for mathematical programming. Though first programming experiences will necessarily be simple, even young children can encounter key elements of computational thinking – expressiveness of a "live" language, a drive toward abstraction, simple iteration, and more – in their mathematical learning, supporting the mathematics and becoming a foundation for later years' learning of more sophisticated programming techniques, with consequently increasingly varied applications, as they need them.

« 57 » We have created a sequence of microworlds (and continue to create more) – each comprised of a limited command-set in Snap*!* and a set of puzzles to solve (some purely exploratory, some narrowly focused) through programming. Over the course of a year, children encounter four to six of these microworlds, each designed to support, enhance and extend one or more mathematical topics and practices of their grade.

« 58 » One of our microworlds displays a number line, optionally settable for any range depending on the grade level, purpose, and accompanying puzzle. The ticks mark regular intervals, but interval size is completely settable (consecutive integers, consecutive eighths, skip counting by any amount, starting at any arbitrary number). For the seven-year-olds, the ticks identify consecutive integers, and only one number (usually 0) is labeled, intentionally chosen not to be the leftmost mark on the line (Figure 16).

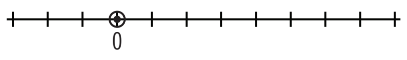


Figure 16 • A number line with ticks representing consecutive integers.

« 59 » The seven-year-olds have a palette of programming blocks, initially just the ones shown in Figure 17. Clicking a block performs the indicated arithmetic, shows the corresponding movement on the line, and

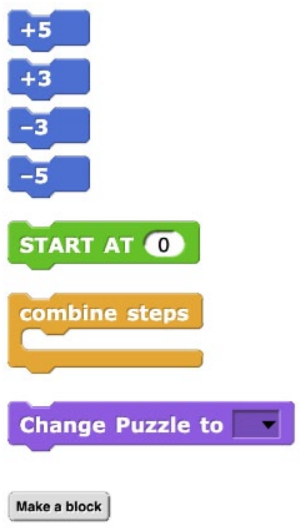


Figure 17 • The initial programming blocks for the number line microworld.

8| A not uncommon six-year-old's pronunciation of "eighth."

labels the result. For example, clicking the +5 block moves the sprite 5 spaces right (arc default, but optional) and marks the new number. If the sprite had already been moved to 3, we see the display shown in Figure 18.

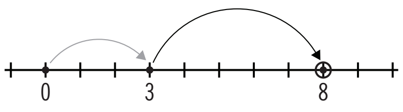


Figure 18 • A move of +5 from 3 to 8.

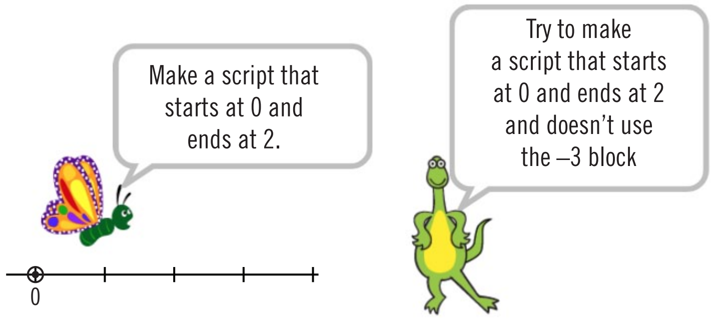

Make a script that starts at 0 and ends at 2.
Try to make a script that starts at 0 and ends at 2 and doesn't use the –3 block
0

Figure 20 • Two programming puzzles.

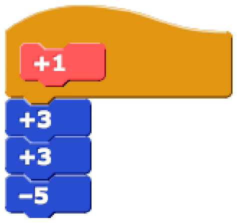


Figure 21 • Definition of the +1 block.

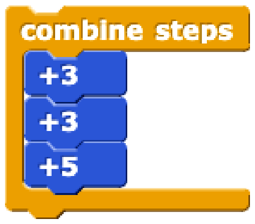


Figure 22 • The combine steps block performs the arithmetic before drawing arcs.

« 60 » Two more blocks let them clear the line and restart at some number (defaulting to 0) and let them choose the puzzle they want to work on. They also get two buttons: one lets them save their work, and one lets them make their own new block. Of course, they may also just play with the blocks they have, without aiming to solve a puzzle.

« 61 » Children explore the tools with very open puzzles like "How many of the numbers from 1 to 10 can you label?" Many initially experiment with no particular plan, but several of the children became interested in the pattern produced by the arrows, and tried systematically to label all the numbers in a "pretty" way.

« 62 » Olivia, a young seven-year-old, created the script shown in Figure 19, and explained how she solved the puzzle. She said "I just went plus 3, plus 3, minus 5. Then, if I click again, it's 2. I just click it fifteen times." Nobody asked why "fifteen." Five clicks will do. This is a wonderful informal example of reasoning by mathematical induction, from a seven-year-old!

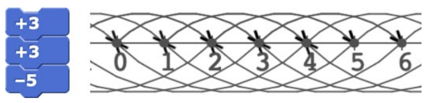


Figure 19 • Olivia's algorithm for getting from 0 to 1 (left) and the pattern it drew after many uses.

« 63 » When the children have learned how blocks can be snapped together to create a script, more focused puzzles of increasing challenge require them to experiment, plan, predict results based on mental arithmetic and even *explain* results. Two puzzles are shown in Figure 20, as they appear to children.

« 64 » Jake asked if they could *make* a block. Yes! We illustrated with Olivia's script. Just click the MAKE A BLOCK button, name the new block – in this case, they named it "+1" because that is what it was intended to be – and drag in the script that made it work (Figure 21). The result was a new block +1 that they can use.

« 65 » Later, another given block allows children to combine steps before (or without) creating a new block. Instead of drawing separate arrows for each of the three steps in Olivia's algorithm, the combined script (Figure 22) shows only the resulting arrow, a single +1 arrow from one number to the next. The abstraction serves both mathematical and CS/CT goals.

« 66 » Some puzzles ask for two different scripts that do the same thing. As it turns out, Olivia's script and her explanation of it solved two advanced puzzles that the class had not yet encountered: one asks for a script that moves from 0 to 1; the other asks children to analyze two scripts (Olivia's and another) and explain why they do the same thing. Teachers can hold class discussions to analyze and explain why a script does what it does, or to predict a result that is not visible on the segment of the number line that they see. For example, shown a script that moves from 0 to 1, one puzzle asks children to "predict where these scripts will land if you start at 19," a number that does not appear on their screen. And then there are proof-challenges "Is there a way to move from 0 to 1 in exactly 2 moves?" or "What is the shortest script that…?"

« 67 » Children routinely visit negative numbers, often by accident, but sometimes on purpose, and always with no fuss and no fanfare. *Many* children have heard of them and are fascinated by them; most children get excited, announce these events, and otherwise ignore them and move on. A few ask questions, and the simple answer is "You know how to get back to the positive side if you want to." This does not obligate any explaining or "teaching" about negative numbers; negative numbers are not in the early grades' curriculum, but the experience is valuable (and builds some correct intuitive ideas) before formality is mandated.

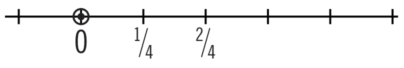


Figure 23 • Zooming in on the number line.

« 68 » Because puzzles like these "have legs" mathematically, they can grow with the child and serve learning in later grades. At the simplest, the very same puzzle set can be used on a "zoomed-in" view of the number line, to explore fractions (Figure 23). In place of +3, −5, etc., children now puzzle with + ¾, − ¾, etc., with similar puzzles challenging them to mark $^1/_4$, $^2/_4$…. If these puzzles feel so familiar as to be "trivial," that's part of the point; these new numbers, fractions, behave in the familiar way because they are just numbers; $^3/_4 + {}^3/_4$ gives $^6/_4$, not the canonically wrong $^6/_8$.

« 69 » Another variant changes the available blocks to ±6 and ±9 and challenges children to label all the numbers they can. Both show how versatile this one puzzle format is, capable of addressing later grade-level standards (e.g., fractions, factors, multiples, common factors, analyzing patterns, building fluency with multiplication facts) and foreshadowing in grade-appropriate ways ideas children will make explicit later.

« 70 » In the same way, students build and compose functions. The seven-year-olds have a set of function blocks for ±1, ±10, and ±100, each with an input slot like this + 10. Children can type a number in the slot 58 + 10 and then when they click on the block, it performs the operation, e.g., 58 + 10 68 or 57 − 1 56. They can also compose functions by dragging one into the input slot of another + 1 + 1. This two-step process adds 1, then adds 1 to the result. Again, children can capture that process and give it a name to say what it does 27 + 2 29. Also, as before, the familiar dinosaur and butterfly offer puzzles: Can you make a block that adds 200? That adds 0? That adds 9? That adds 99? That adds 19? That subtracts 2? That adds 8? That subtracts 9? The last often turns out to be significantly more challenging than the others.

« 71 » Standard approaches push for automaticity at adding or subtracting 9 and 8 to single-digit numbers, relying on paper-pencil algorithms for larger computations. But by seeing these operations as easy, automatic two-step algorithms, students can perform the same calculations with *any* number, an idea that generalizes to many other approximate-then-adjust approaches.

« 72 » In later grades, children encounter essentially the same idea, and comparable puzzles, with multiplication and division. They start with blocks ×2, ÷2, ×10, ÷10, ×100, ÷100, and can compose operations like 86 x 10 860 and 86 ÷ 2 x 10 430 into two-step algorithms like 86 x 10 ÷2 430 86 ÷ 2 x 10 430. Comparing such two-step processes to one-step operations they already know lets them produce new blocks, like 86 x 5 430.

« 73 » With verbal (not written!) practice structured to take advantage of our "built-in" cognitive expectation of the distributive property and of the linguistic relatedness of, for example, *six*, *sixty*, *six-hundred*, it is relatively little work for (most) children to learn to halve any number mentally. The pattern of multiplying by 10 is even easier to acquire. (The reason why the "tack-on-a-zero" pattern works is often harder to grasp, but worth building.) In any event, for many children who have built those two skills, the experience of inventing and building a ×5 machine as either ÷ 2 x 10 or x 10 ÷2 lets them become quite adept at mentally multiplying *any* two-digit numbers by 5, supplementing the one-digit facts that their teachers and parents want them to acquire. Again, the puzzles ask them to invent a variety of new tools like ×4, ÷4, ×100, and so on. And, again, they play.

« 74 » It is clear where these puzzles are going *mathematically*, but where are they going *creatively*? Let us look again at the sense in which these are "puzzles" and not just standard exercises.

« 75 » In the earliest puzzles – composing ±1, ±10, and ±100 blocks to build new blocks like 58 + 9 67 – the children create many special blocks themselves, mastering the *reasoning*: composition of mentally easy and understandable place-value-based operations to do more "difficult" operations. They are learning not just a specialized trick or two but a way of thinking, a way to invent mathematical methods. We pose only a limited set of puzzles, both because we cannot think of all possibilities and also because there is no need to; we deliberately leave room for the children to play. And they do play. A lot. Nobody chooses (or sticks with) play that bores or defeats them, so the children create their own differentiated learning. The challenges they create for themselves are (generally) precisely at their own frontiers of knowledge, skill, cognition, and interest in ways we could not have known. Part of this readiness to play appears to be the direct result of having a notation system (programming) that is active, unlike marks on a paper that just sit there passively. They treat these tasks as puzzles, trying to see – just as they might on a playground – what new trick they can do. Standard math problems are "done" as soon as one has written a number on the page.

« 76 » Of course, mathematics is more than arithmetic, so an approach that uses programming as an expressive language to support mathematical learning must provide vocabulary and methods for handling shape, size, angle, distance, structure…, and good situations – puzzles – in which to explore those ideas. ScratchMaths gives a beautiful example of a focus on angle, distance, and structure – structure in the code itself as well as in the visual, often symmetric, designs it produced. Angle is subtle in many ways – not just the conventions for quantifying angles in degrees and the modularity of that quantification, but also just the multiple meanings and images of angle – and consequently hard to present in a clean way to seven-year-olds. But young children *can* create code that navigates a map and they particularly happily play with puzzles involving distance and direction on simple grid-like maps of "towns" in which various buildings (houses, schools, libraries) are personalized with the children's own names. ("You've found paths between Mia's and Adam's houses that are four blocks long. You've found longer paths that are six blocks long. Can you find a path that is exactly five blocks long?")

« 77 » Producing and interpreting small arrays as images of multiplication is mandated mathematical content, and the relevant puzzles can be fun and attractive. Children generate colorful rows (solid or patterned) of repeated squares, and arrays from repeated rows, and the similarities of the algorithms inside draw row and draw array illustrate the meaning *and value* of "abstraction." Abstraction includes both generality and "hiding complexity" – suppressing details or identifying the important characteristics for a particular purpose – by creating a single new command/function to replace a longer collection of instructions that would otherwise have to appear in several places. Older children may parameterize their blocks that draw rows or arrays. A `draw array` block depends on two parameters, the dimensions of the array; a `draw rectangle` block is a further (simpler) abstraction, using the same two inputs but drawing only the border of the array. For either of these, older children might invent a playful quiz, having their program draw a random-sized array and ask about area (how many tiles it has) or perimeter, using their own reasoning about those inputs in order to teach the program how to calculate the correct answer.

« 78 » At the high-school level, programming allows students to *build* the mathematical objects and processes that they are studying: relatively easily, they can build functions that manipulate polynomials, transform points with matrices, render a set of points in space in a convincing projection on the screen (Lewis 1990), and study algebraic structures (Cuoco 1990). And tools such as Geometer's Sketchpad or Cabri – not programming as it is usually thought of, but programming nonetheless, with construction specified by the students rather than just use of the computer to manipulate predesigned models – allow students studying geometry to build models of mathematical objects and ideas, and to explore the consequences of manipulations of those models.

Programming in general

« 79 » The current excitement with "coding for all" creates a challenge. There is no more room in the curriculum. If coding (for coding's sake) is added, what gets shoved out of the way? But if coding is learned *in service of content that is already core*, it is not a displacement. Our motivation for programming in elementary-school *mathematics* was for the sake of the mathematics – *not* an "extra," but an improvement of content that is already core. It *also* serves the push for "coding." Mathematics is not the only core context in which programming could potentially serve as a supportive, non-distracting medium but, at the elementary-school level – especially in the early grades – it may be the easiest and most natural. And what seven-year-olds can do allows eight-year-olds to do more. Incrementally, it sets a strong foundation for secondary students' learning.

« 80 » Moreover, genuine continuity can be achieved. Beauty and Joy of Computing (BJC)[9] is an entirely separate piece of work, an Advanced Placement Computer Science course whose explicit mission is broadening participation in computer science. In service of this goal, BJC takes on computer science with a programming-centric approach, letting students experience the joy of creation and see beauty not only in the objects they can produce through programming, but also in the programs themselves. It introduces the elegance of recursion and higher-order functions, making these reputedly "difficult" topics accessible by virtue of the lucid visual imagery of Snap*!*, a language that is not unreasonably characterized as Scheme disguised as Scratch.

« 81 » Initial funding for BJC required it to be an Advanced Placement course with a framework dictated by the College Board. Even so, except as constrained by AP requirements, BJC is largely project-based with experience before formality; the explorations through which programming is learned include projects set in contexts like art and graphics, linguistics, mathematics, and games. While BJC is not at all a math course, its activities naturally touch – and help teach – many conventional mathematical content topics, and its approach to programming is consistently focused on mathematical and computational thinking (CT). The reason it introduces various contexts – the arts, linguistics, etc. – is partly to meet the varied interests of students, but much more to show how broadly programming applies, how broadly the students can allow their ideas and creativity to wander, how much they can tailor their own projects, for which the AP framework allocates time, in their own personal direction.

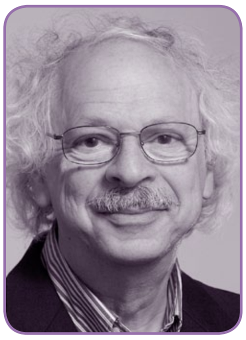
« 82 » Even though BJC is explicitly an AP course for high school, excerpts *involving recursion* were used successfully in a computer science elective with sixth graders. They wrote recursive code to draw a complex tree, and here they and their teacher are giggling at the result of a gossip-producing program with a randomly invoked recursive step that, in this case, generated a very long sentence (Figure 24). Other students in this elective created a program to conjugate Spanish verbs properly so that they could generate sentences in Spanish. They tested the work of their programs by using `map`, a higher-order function, to apply their conjugation block to a list of verbs.

9| http://bjc.edc.org

Figure 24 • Surprise and delight at the complex result of a recursive process.

Playgrounds

« 83 » Giving even very young students a way to think algebraically using bags and marbles lets them invent mathematical tricks they love. It prepares them for algebra but more importantly, it lets them feel smart and pose problems and *play* with their own algebraic ideas. More broadly, treating mathematics as serious intellectual play, puzzling things out by searching and researching, and gaining the intellectual tools for posing one's own challenges teaches children to be mathematicians. Papert suggested programming as a medium for that, but the essential ingredient remains the promotion of serious intellectual play. Programming taught just as a skill or to meet new standards may well not serve that purpose. But if a programming environment lets students explore and create, provides good tools for doing that, and gives students the "third language of mathematics" so that as their ideas and thinking grow in sophistication they have a language for expressing and honing those ideas, such an environment does add a new playground consistent with Papert's vision of children being creatively engaged as mathematical thinkers.



PAUL GOLDENBERG
has been at EDC for over 30 years. He has taught primary-school, middle-school, high-school computer science, and graduate-school mathematics and psychology for education. He worked at the MIT Logo Laboratory with Seymour Papert and at BBN Labs with Wallace Feurzeig. At EDC, he designs, crafts, and researches learning materials for K-12 students and teachers, using or rekindling their natural curiosity about and interest in mathematics. With many others, Paul helps lead EDC's initiatives to broaden access to CS. The NSF funded BJC curriculum emphasizes programming and social issues of computing, and is now taught to over 2,500 students/year in New York City alone. See Paul's blog posts about early mathematics: http://ltd.edc.org/wrong-answers-or-wrong-questions and https://blogs.ams.org/matheducation/2018/09/02/ideas-under-construction-children-saying-what-they-know/.

Conclusion

« 84 » A few states, including Massachusetts (where I live), have begun to develop frameworks for CT across the grades (http://www.doe.mass.edu/frameworks/dlcs.pdf). CT is variously defined but always includes elements like abstraction, algorithm, modeling and simulation, programming, and data (with an implication, not reflected in all implementations, that "data" means *big* data). Not surprisingly, to help develop this thinking there has been a proliferation of on-computer activities not involving programming and also "unplugged" activities to the same end.[10] The difficulty of adding anything to an already jam-packed school day has led to much talk about integrating CT activities into existing content areas, particularly science and mathematics (e.g., https://go.edc.org/elementary-ct), but also language. In my opinion, some of the integration suggestions are shallow, but that should be no surprise at a time when the whole effort is so new.[11] Still it got me to thinking about why my own inclination has been *toward* programming, not away, and toward abstraction, and algorithm rather than modeling and simulation, whenever the aim is explicitly to integrate with other subjects.

« 85 » I think my particular leaning may be largely bias, possibly the result of my greater focus on elementary and middle school, and greater focus on mathematics than on science. At the elementary-school level, modeling and simulation seem easier to integrate with science than with mathematics; programming, along with abstraction and algorithm, seems easier to integrate with mathematics than with science.

« 86 » Modeling, for example, is something that mathematics (and mathematicians) can *do*, and since mathematics can build models of mathematical ideas, modeling is also something that mathematics *uses*. But, at least as far as I see at the elementary-school level (especially in the early grades) modeling *with* mathematics – creating mathematical models of phenomena – is very limited. And it is fairly abstruse, in the following sense. While *every* mathematical statement (like "there are seven cows") is an example of an abstraction (the cowness is reduced to irrelevancy) and just a model of the experiential reality, no child in the known universe thinks of such a statement as an abstraction or a model. That level of abstraction is so normal to them that it is totally "invisible" – it is just what language does. By contrast, modeling is a natural place to focus in science – the core of experimentation and the form of many scientific claims – and simulation (at least as generally used) is an automation/extension/elaboration of modeling.

« 87 » Programming is exactly the opposite, easier to integrate into (early) mathematics than into science. (Of course, take this with a grain of salt, as I have not given scientific programming nearly as much thought. As I advertised, these are wild final thoughts that I might disown tomorrow.) That may be partly because the kinds of statements one makes in early mathematics tend to be about relationships and about simple processes. "Writing a program" that enacts a function, like doubling or adding 10 to its input, is easy programming. Indeed, it is easier to write in a general way as a program (a Snap! block) than as a paper-pencil scrawl, because a program is an *active* notation; it will *perform* the action and give feedback, which paper-pencil scrawls do not. It is also a structured notation, imposing a bit of order on what young students typically scatter over a page in a way that, even if totally correct, does not reveal their logic. Similarly, writing a program that pairs elements of two sets, writing a program that draws simple shapes, or creates arrays or paths to study, is mathematically on task and easy programming. By contrast, most scientific phenomena are too complex for young children to model by writing a program (often pretty complex even for adults).

« 88 » I would love to get reactions to this last, very spur-of-the-moment rumination. What genuine *programming* activities, *at the elementary-school level*, can be integrated with science in a developmentally appropriate and scientifically relevant way? And what *modeling* or *simulation* activities, again at the elementary-school level, can be integrated sensibly with mathematics?

10| As I was completing this article, I received a copy of *Bebras* (http://www.bebras.lt), a set of activities, many puzzle-like, that I found quite appealing, all designed to develop various elements of CT in students.

11| And, clearly I, myself, am being a bit shallow in using the vague quantifier "some suggestions." Of course, in *any* situation, *some* suggestions will be shallow.

Acknowledgements

Funding for doing and reporting the work described in this article was provided in part by the National Science Foundation, grants 0917958, 1135173, 1441075, 1543136, 1621011 and 1741792. Views expressed here are those of the author and do not necessarily reflect the views of the Foundation.

References

Cuoco A. (1990) Investigations in algebra. MIT Press, Cambridge MA.
Danesi M. (2002) The puzzle instinct. Indiana University Press: Bloomington IN.
Goldenberg E. P. & Carter C. J. (2018) Myths of priority and unity in mathematics learning. Education Sciences 8(2): 85. https://www.mdpi.com/2227-7102/8/2/85/htm
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, Portsmouth NH.
Goldenberg E. P. & Shteingold N. (2007a) Early algebra: The MW perspective. In: Kaput J. J., Carraher D. W. & Blanton M. L. (eds.) Algebra in the early grades. Erlbaum, Hillsdale NJ: 449–475.
Goldenberg E. P. & Shteingold N. (2007b) The case of Think Math! In: Hirsch C. (ed.) Perspectives on the design and development of school mathematics curricula. National Council of Teachers of Mathematics, Reston VA: 49–64.
Lewis P. (1990) Approaching precalculus mathematics discretely. MIT Press, Cambridge MA.
Mark J., Goldenberg E. P., Kang J., Fries M. & Cordner T. (2014) Transition to algebra. Heinemann, Portsmouth NH.
Otten M., Heuvel-Panhuizen, M van den, Veldhuis M., Heinze A. & Goldenberg E. P. (2017) Eliciting algebraic reasoning with hanging mobiles. Australian Primary Mathematics Classroom 22(3): 14–19.
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.
Sawyer W. W. (2003) Vision in elementary mathematics. Dover, New York.
Schulz L. E. & Bonawitz E. B. (2007) Serious fun: Preschoolers engage in more exploratory play when evidence is confounded. Developmental Psychology 43(4): 1045–1050.
Sendova E. (2013) Assisting the art of discovery at school age: The Bulgarian experience. In: Sanchez-Escobedo P. (ed.) Talent development around the world. Mérida, Yucatán: 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.
Wirtz R., Botel M., Beberman M. & Sawyer W. W. (1964) Math workshop. 17 volumes. Encyclopaedia Britannica Press, Chicago IL.

Received: 7 January 2019
Accepted: 28 March 2019

Open Peer Commentaries

on Paul Goldenberg's "Problem Posing and Creativity in Elementary-School Mathematics"

Keeping the Children as Question Marks: Educational Attempts to Tap Curiosity and the Drive for Challenge

Evgenia Sendova
Bulgarian Academy of Sciences, Bulgaria
jenny.sendova/at/gmail.com

Pavel Boytchev
Sofia University, Bulgaria
boytchev/at/fmi.uni-sofia.bg

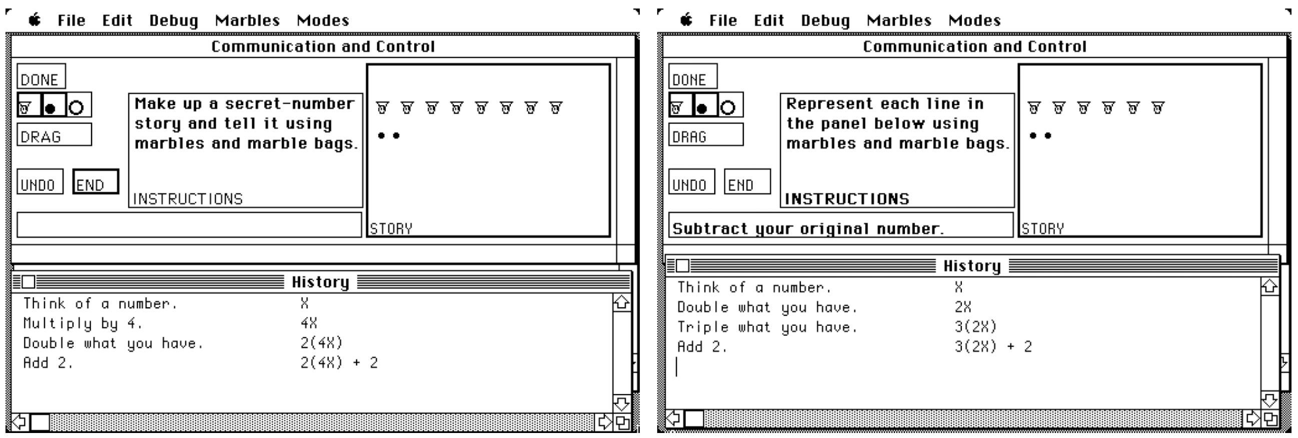
> Abstract • Supporting the inborn curiosity of children is the motivation for our involvement in developing novel curricula, textbooks and microworlds. Our main goal of implementing the constructionism as a fundamental educational strategy is to keep the students "as question marks," i.e., to encourage them to pose questions, to make experiments, to invent their own problems. We strongly support the ideas behind Goldenberg's experience in learning environments, generating curiosity and creative engagement (§15). As an extension of the ideas in §54 we propose a metaphor to visualize how programming can be "repurposed" to *wrap the math* in an attractive, yet educationally effective way.

"Children enter school as question marks and leave as periods."
(Postman & Weingartner 1969: 53)

Can you solve my problem? – Supporting students to invent their own problems

« 1 » In 1999 Seymour Papert formulated the *Eight Big Ideas Behind the Constructionist Learning Lab.*[1] Two of them are *hard fun* and *taking time*. The problem with implementing them in the regular school setting is that it "is even more rigidly constrained than it used to be," as Paul Goldenberg states in §1 of his target article. It

1| https://inventtolearn.com/8-big-ideas-of-the-constructionist-learning-lab/


File Edit Debug Marbles Modes
Communication and Control
DONE
DRAG
UNDO
END
INSTRUCTIONS
STORY
History
File Edit Debug Marbles Modes
Communication and Control
DONE
DRAG
UNDO
END
INSTRUCTIONS
STORY
History

Make up a secret-number story and tell it using marbles and marble bags.

```
Think of a number.            X
Multiply by 4.                4X
Double what you have.         2(4X)
Add 2.                        2(4X) + 2
```

Represent each line in the panel below using marbles and marble bags.

Subtract your original number.

```
Think of a number.            X
Double what you have.         2X
Triple what you have.         3(2X)
Add 2.                        3(2X) + 2
```

Figure 1 • Screenshots of the work of Thompson (1989) with a prototype of the software developed by Feurzeig. Left: Creating a secret number story, translated in English and in algebraic notation; Right: Using virtual marbles and marble bags to represent each line in the story.

is not easy to evaluate the achievements of students and teachers, creatively engaged in mathematical activities, especially if assessment instruments are focused on multiple-choice tests. Still, Goldenberg demonstrates how children in the (pre-)primary school can be supported in solving and creating math problems. This requires dedicated teachers who enjoy challenges posed by children and look for children's reasoning.

« 2 » Goldenberg shares his fascination with an "algebraic approach to teaching elementary arithmetic" in which the emphasis is on "play and surprise" (§12). Although the content corresponds to the level expected for the grade, it challenges the children to do research, to observe patterns, to figure out why a puzzle works, to create ones of their own. This serves as a propaedeutic for the formal algebraic language they will learn later.

« 3 » In this context the reward is not to get a single correct answer. To see the generality, "to create that abstraction for themselves, children need research time" (§30) – to illustrate the instructions of the math trick by their own pictures, to try the trick with different numbers so as to extract the general rule, and finally to figure out how to construct their own tricks (§§28–48).

« 4 » The technique of interpreting linear equations with bags and marbles is implemented in various virtual environments providing students with platforms for understanding ideas behind formal manipulations. A focus on algebra as a language for describing relationships of quantities manifests itself in the computer environment *Marble Bag Microworld*, developed by Wally Feurzeig (1986) as a machine implementation of the idea presented in 1964 by Sawyer (2003) and implemented in Wirtz et al. (1964) (§12). Students are introduced to standard algebraic notation by creating and solving story problems. They observe the correspondence between the iconic, English, and standard algebraic representations (Figure 1). The goal is that "these activities provide a cognitive foundation for students' understanding of operations on equations" (Thomson 1989: 13).

« 5 » An example of the work of *Equation Balance* by Pavel Boytchev is shown in Figure 2. It features scales with golden bars (units) and boxes (variables) representing a linear equation. The initial problem can be solved in different ways, and then the solution is used as a generator of new problems.

« 6 » Tom McDougal, a math teacher from Chicago, extended *Equation Balance* to allow variables on both sides, negative values and custom equations. He described the usage of the application with eighth-graders as "fabulously effective" (Boytchev 2019).

« 7 » An interesting question considered in Goldenberg's §66 relates to how children explain different scripts doing the same thing. We arrived at such a situation in the Weblabs project (Gachev, Sendova & Nikolova 2005). Sixth-graders from Bulgaria and Portugal had created ToonTalk robots producing seemingly the same infinite sequence. After comparing the algebraic representations and the Logo procedures the final proof of equivalence was given by means of the difference equations theory. All the problems created by children presented in the target article have arithmetic (later algebraic) content. As far as geometric content is concerned, such experience is very limited. In Bulgaria, stereometry is introduced in the fifth and sixth grades, the focus being on learning the formulae rather than on stimulating spatial imagination, or on formulating problems. An approach to improving the situation was offered by the DALEST project (Developing an Active Learning Environment for the Learning of Stereometry) (Boytchev, Chehlarova & Sendova 2007). Students from five countries were provided with applications to explore the properties of 3D objects, to solve and formulate their own problems. Here is an example of a problem created by Koya, a 12-year-old girl (Chehlarova & Sendova 2009):

"Koya's problem: Eliminate a cube from each of the compositions in Figure 3 so as to get respective compositions for which each layer (along each direction) contains a cube of each color present in the composition."

« 8 » The steps Koya took in the process of creating a problem were verbalizing the idea to use a two-color composition of size 3x3x3; adding additional cubes to generalize the problem with three colors; and tuning the formulation to a language closer to that of everyday life. These steps were described in a DALEST scenario to be used by other students for creating their own problems.

« 9 » What we observed in children's work was that they were eager to show their problems to peers and observe their reactions. The children shared the way of composing the problem, discussed some errors, and modified the original composition into new problems.

Happy Birthday, Miss Pencheva! – Inventing problems as presents

« 10 » An interesting series of logical problems appeared as a collective present to Galya Pencheva, a dedicated math teacher and graduate student at IMI-BAS. Although she had given students the task of inventing their own problems on a specific topic, the final booklet of problems from her fourth-graders came as a total surprise. Figure 4 shows one of them.

« 11 » In a nutshell, the importance of cultivating the skill in children of asking questions, and posing and formulating problems should be recognized by teachers and teacher educators and supported by the designers of learning environments. And the novel learning environments (digital and non-digital alike) would gain from "stepping on the shoulders" of educational giants such as Sawyer (§12).

The centrality of question asking

« 12 » Stephen Brown and Marion Walter (2005: 3) state,

"[t]he centrality of problem posing or question asking is picked up by Stephen Toulmin in his effort to understand how disciplines are subdivided in sciences."

They refer to this passage in Toulmin:

"If we mark the sciences off from one another […] by their respective 'domains,' even these domains have to be identified not by the types of objects with which they deal, but rather by the questions which arise about them […]" (Toulmin 1977: 149)

« 13 » Studying disciplines separately simplifies things, allows us to stay focused and is useful in the short term, but in the

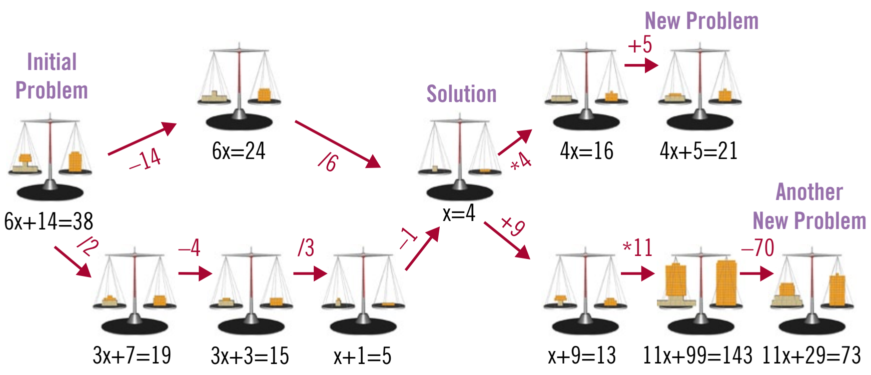


Figure 2 • Equation Balance – problem solver and problem generator.

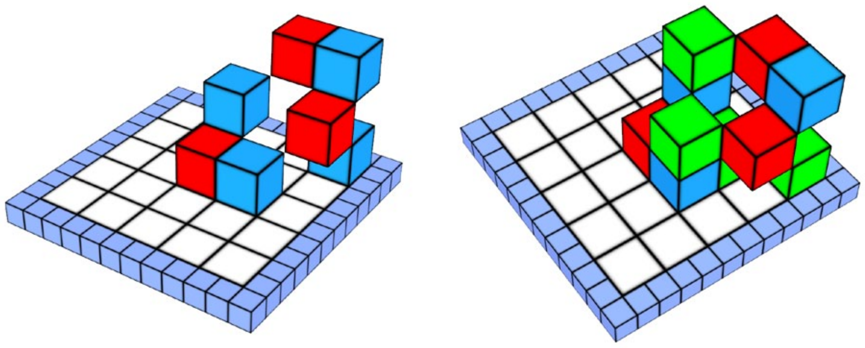
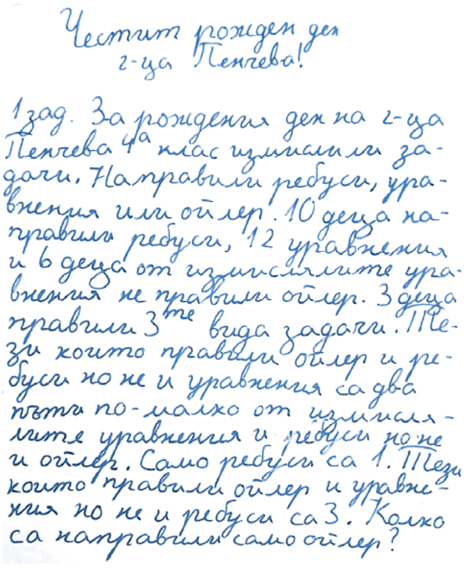


Figure 3 • Compositions with two and three colors.



Честит рожден ден
г-ца Пенчева!

1 зад. За рождения ден на г-ца
Пенчева $4^{а}$ клас измислили за-
дачи. Направили ребуси, ура-
внения или ойлер. 10 деца на-
правили ребуси, 12 уравнения
и 6 деца от измислилите ура-
внения не правили ойлер. 3 деца
правили $3^{те}$ вида задачи. Те-
зи които правили ойлер и ре-
буси но не и уравнения са два
пъти по-малко от измисля-
лите уравнения и ребуси но не
и ойлер. Само ребуси са 1. Тези
които правили ойлер и уравне-
ния но не и ребуси са 3. Колко
са направили само ойлер?

The 4th-graders taught by Miss Pencheva decided to create problems as a present for her birthday. They made rebuses, equations and diagrams. 10 children made rebuses, 12 made equations and 6 of those who created equations had not made diagrams. 3 children created problems of each of the 3 types. Those who invented problems on diagrams and rebuses but not on equations were half as many in number as those who invented equations and rebuses but not diagrams. There was 1 child who invented only rebuses. Those who created problems on diagrams and equations but not on rebuses numbered 3. How many children created only problems on diagrams?

Figure 4 • Left: The original formulation of one of the children's greeting problems. Right: Translation.

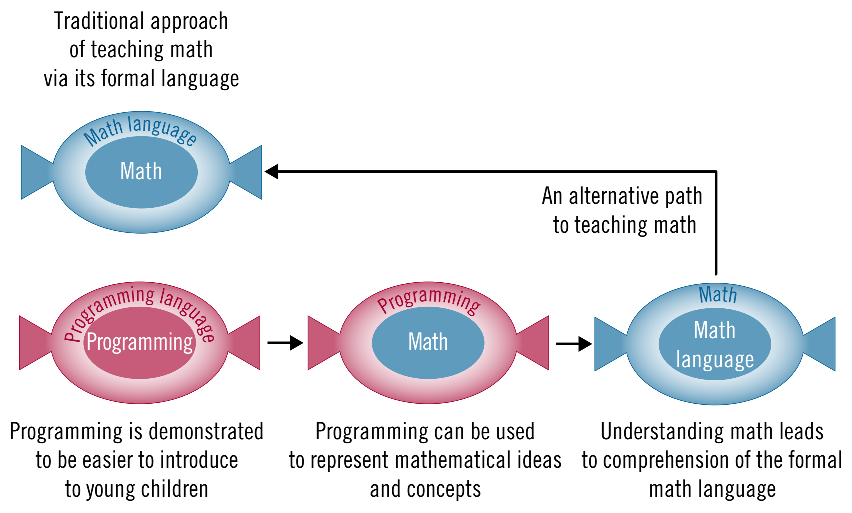


Figure 5 • Using programming to learn Math and Math language.

long term – it is devastating. This is something we observe in our work with undergraduates and with in-service and pre-service teachers. They find it difficult to apply their knowledge from one field to another, and even worse – they cannot imagine that they could do it. After discussing how to visualize the problem, one of us came up with the following metaphor.

Boytchev's metaphor

« 14 » Every science consists of a kernel (*candy*) and an interface (*wrapper*). In mathematics the kernel is the genuine mathematical knowledge – mathematical notions, phenomena, models, etc., whereas the interface is the mathematical language to access and express the kernel – formulae, variables, etc.

« 15 » Similarly, the sorting algorithms belong to the kernel of programming, while the programming languages are the interface. Every science has its specific, unique interface, but their kernels share a lot of common features. The appearance of multidisciplinary sciences (bioinformatics, astrophysics, computational neuroscience, etc.) is an attempt to use the interface of one science with the kernel of another – this leads to new ideas and discoveries.

« 16 » The problem with working with younger children is that it is difficult for them to *unwrap the candy*. If the kernel is in a child-friendly wrapper, they will be eager and will be able to unwrap it. We find the same idea in Goldenberg's §54. Programming can be "repurposed" to wrap the math in an attractive, yet educationally effective way. It is not used to replace the formal interface, but is used to reach the core and eventually to master the formal math language later on (Figure 5).

« 17 » In response to Goldenberg's §88 we would paraphrase Richard Feynman (1999: 4): that even though we do not know the answer we find it optimistic *to keep trying new solutions* since this *is the way to do everything*.

References

Boytchev P. (2019) Developing educational content [in Bulgarian]. University Press "St. Kliment Ohridski", Sofia (in press).

Boytchev P., Chehlarova T. & Sendova E. (2007) Virtual reality vs real virtuality in mathematics teaching and learning. In: Proceedings of the Joint IFIP Conference Informatics, Mathematics, and ICT: A "golden triangle" (IMICT 2007), Boston, USA. CCIS, Northeastern University, Boston: 1.

Brown S. I. & Walter M. (2005) The art of problem posing. Third edition. Lawrence Erlbaum Associates, Mahwah NJ.

Chehlarova T. & Sendova E. (2009) Enhancing the art of problem posing in a dynamic 3D computer environment. In: Proceedings of the 6th congress of the mathematicians of Republic of Macedonia, Struga 19–22 October 2008. Alfa 94 MA, Skopje: 19–28.

Feurzeig W. (1986) Algebra slaves and agents in a Logo-based mathematics curriculum. Instructional Science 14(3–4): 229–254.

Feynman R. (1999) The meaning of it all. Penguin, London.

Gachev G., Sendova E., Nikolova I. (2005) The-more-it-changes-the-samer-it-gets principle in the context of mathematics and informatics education. In: Gregorczyk G., Walat A., Kranas W. & Borowiecki M. (eds.) Proceedings of EUROLOGO '2005. DruksFera, Warsaw: 87–99.

Postman N. & Weingartner C. (1969) Teaching as a subversive activity. Dell, New York

Sawyer W. W. (2003) Vision in elementary mathematics. Dover, New York.

Thompson P. W. (1989) Artificial intelligence, advanced technology, and learning and teaching algebra. In: Kieran C. & Wagner S. (eds.) Research issues in the learning and teaching of algebra. Erlbaum, Hillsdale NJ: 135–161.

Toulmin S. (1977) Human understanding. Princeton University Press, Princeton NJ.

Evgenia Sendova is an associate member of IMI-BAS, at the Education in Mathematics and Informatics Department. As a member of an educational experiment (1978–1999) conducted by BAS and the Ministry of Education, she was involved in creating a new curriculum, educating teachers and students. Since 1997 she has been a tutor at RSI – an international summer program for high-school students organized by the Center of Excellence in Education and MIT, USA. Sendova is currently the IMI-BAS coordinator of the Scientix European project.

Pavel Boytchev is an associate professor and researcher at the Faculty of Mathematics and Informatics, Sofia University. His research interests are developing courses and educational software based on computer graphics. He has created a dozen courses, hundreds of computer-generated video clips and thousands of computer programs.

Received: 27 May 2019
Accepted: 10 June 2019

Problem Posing and Programming as a General Approach to Foster the Learning of Mathematics

Mattia Monga
Università degli Studi di Milano, Italy
mattia.monga/at/unimi.it

> Abstract • Finding tasks to propose to children, able to generate curiosity and creative engagement without having just gamified school exercises is very difficult. The appeal of a traditional curriculum is its (deceptive) scalability. Programming as a language is a powerful metaphor, much more powerful than the one suggested by using simple, predefined building blocks that fit well together.

Creativity, puzzles, and educational agendas

« 1 » In the target article Paul Goldenberg argues that puzzles have the potential to engage children in a genuine mathematical research activity, even more so when they are requested to produce variants of the puzzles they solved (§§1–50). Goldenberg shows interesting examples that are both surprising and generate curiosity (as promised in §16). However, from my experience with the organization of a computational thinking contest in which every year we propose tasks to students with similar goals to those in Goldenberg's scheme, I learned that it is not easy to keep the bar of surprise and curiosity high while pursuing an educational agenda. If one wants to cover a specific curriculum, it can be hard to come out with clever puzzles instead of trivial gamified exercises unable to fully engage the creative impetus of children.

Puzzles and the appeal of abstract thinking

« 2 » Mathematical puzzles have a very long history: several examples from Greek or Vedic mathematics are well known and they amused generations of curious students. They were probably invented to summarize or exercise mathematical knowledge, but also to make the abstract nature of mathematics more palatable to the general public. This is why we found them not only in the mathematical literature, but also in writings directed to a wider audience. For example, the well-known Diophantus' epitaph:

"Here lies Diophantus," the wonder behold.
Through art algebraic, the stone tells how old:
"God gave him his boyhood one-sixth of his life,
One twelfth more as youth while whiskers grew rife;
And then yet one-seventh ere marriage begun;
In five years there came a bouncing new son.
Alas, the dear child of master and sage
After attaining half the measure of his father's life chill fate took him.
After consoling his fate by the science of numbers for four years, he ended his life."

was not found in a mathematical text, but came to us from a collection of Greek epigrams and short poems (the Anthologia Palatina) directed to a general audience.

« 3 » Today the use of puzzles, riddles and other short challenges to complement or aid the learning of mathematics is popular, as testified by several successful initiatives, able to engage millions of school pupils every year. One of the most widespread is the Kangourou des Mathématiques with about 6,000,000 participants from 78 countries, see http://www.aksf.org/statistics.xhtml. It is a game contest created in 1991 in France by André Deledicq on the model of the Australian Mathematics Competition, with the goal of contributing to the popularisation and the promotion of mathematics among young people with the contest, but also through the associated distribution of a massive and pleasant documentation on mathematics to the participating pupils and their teachers.

« 4 » The Kangourou game-contest idea was brought to informatics and computational thinking by Valentina Dagienė, who in 2004 started the Bebras challenge.[1] Bebras tasks are designed to promote interest in informatics through recreational although educational activities (not necessarily computer-based). Participants are usually supervised by teachers who may integrate the Bebras challenge into their teaching activities.

« 5 » Both Kangourou and Bebras are successful initiatives, at least judging by their popularity among students and, even more, among teachers who see their pupils enthusiastically engaged with the subject they love, compared with the boredom and ineffectiveness generated by traditional approaches. However, finding the right tasks to propose to children, able to "generate curiosity, the creative engagement that Papert referred to as the experience of the mathematician" as Goldenberg writes in §16 of his target article without having just gamified school exercises is very difficult and takes the efforts of a whole research community as varied as Kangourou or Bebras ones. I doubt a single teacher or even a classroom/school chapter of teachers could sustain the necessary creativity for long enough to cover the needs for their curricular programs. Even Bebras delegates often struggle to propose new task ideas and several parts of the area of expertise that Bebras aims to address remain uncovered. It is not easy to categorize either Bebras tasks or the target competences (see Dagienė, Sentance & Stupurienė 2017 and Lonati et al. 2017 for alternatives), but every taxonomy has items that are neglected in most editions and others that repeatedly attract the ideas of task designers every year (although often with simple variants of a main basic idea) even if the role of the issue they cover is relatively marginal in Bebras goals.

« 6 » The appeal of a traditional curriculum is its (deceptive) scalability: writing systematic textbooks with a large collection of repetitive exercises is easier than designing engaging activities for a whole curriculum. And the creative engagement associated with what is successfully learned with well designed "research playgrounds" risks obscuring the value and diverting efforts from more grueling areas. This does not mean that the project-based approach should not be pursued further, but it should remind us that we all "learned" some things by a rote or strictly algorithmic approach, and yet they came out still having a power in our minds.

1 | Bebras is the Lithuanian word for beaver, an animal that is somewhat common in Lithuanian folklore and is iconic of a major computer science problem, the "busy beaver." See https://www.bebras.org About 3,000,000 participants from more than 50 countries took part in 2019.

Programming and learning mathematics

« 7 » Goldenberg argues that programming can become a language for and a natural part of learning mathematics (§51). Coherently with the suggestions of the first part of the article (§§1–50) he reports on a puzzle-based research playground that involves programming. Although some more sophisticated examples are mentioned (§§80–82), the ScratchMaths showcase proposes a programming tool with a limited potential of expression, based only on simple sequences of predefined steps. If programming is supposed to be a key mental tool to express mathematical ideas, this seems to me too limited even at the target age addressed by Goldenberg's proposal.

Programming as a language

« 8 » From a constructionist viewpoint of learning, programming languages may have a major role: in some sense they can be a tool for sharing "artifacts" able to explicitly show one's theories of the world. The crucial part is that artifacts can *be executed* independently of the creator: someone's (coded) mental process can become part of the experience of others, and thus criticized, improved, or adapted to a new project. Papert's experiments with the programming environment LOGO were designed exactly to let pupils tinker with math and geometry.

« 9 » But LOGO is itself a formal language, maybe more regular and simpler than the traditional mathematical one, but still requiring an effort to understand its precise semantics and the "notional machine" that enacts the utterances one invents. In the last decade, a number of block-based programming tools (such as Scratch and Snap!) have been introduced, which should help students to have an easier time when first practicing programming. These tools, however, while reducing the friction with traditional (text-based) syntactic rules can even make the thing more complex when the programmer should focus entirely on the processing of the pieces of information she wants to consider. The color, shape, position on the screen, etc., are all things that could, in principle, be used to change the meaning of a block instruction and the learner may wonder if the semantics of the interpretation depend on these details or whether they can be safely overlooked. In general, visual programming languages do not seem to necessarily help students learn other programming languages (Lewis et al. 2014).

« 10 » Blocks are probably a good choice for the very limited computational variety of the tasks described in Goldenberg's article, in which a fixed sequence of instructions does the trick. The only form of programming abstraction that is introduced is the "naming" of a value or a sequence of instructions.

« 11 » The metaphor of building something by putting together building blocks that fit well together, however, can screen off the epistemic power of language to express problems and suggest solutions, or even make them emerge as in the case of recursive descriptions. Moreover, if programming is intended to be the tool able to make explicit one's mental discourse, a more linguistic metaphor may help. According to Papert,

"in teaching the computer how to think, children embark on an exploration about how they themselves think. The experience can be heady: Thinking about thinking turns every child into an epistemologist, an experience not even shared by most adults." (Papert 1980: 19)

And thoughts made by words and rich grammatical structures are closer to the way knowledge is developed and transmitted between generations, thus something we need to manage to improve ourselves.

« 12 » Expressing something in a way an automatic interpreter can "understand" (without appealing to intuition) can be fruitful for out-of-classroom activities, too. Juggling, for example, can be analyzed with a procedural language: the identification of *proper* sub-activities (i.e., sub-routines like TOP-RIGHT to recognize when one juggling ball is at the top of its trajectory going to the right, or TOSS-LEFT to throw the ball with the left hand) may significantly shorten the time for acquiring juggling skills (from days to hours, according to Papert 1980).

« 13 » In order for this to work, however, some sort of identification with the interpreter is useful. In LOGO (but also in other more recent proposals such as the educational turtle library in Python) the interpreter becomes a "persona," and computation is then carried out through anthropomorphic (or, better, zoomorphic, since animals are very common) actions. In programming, computational processes that evolve in time are described by static texts (or blocks): the mapping between processes and their description is not trivial and it requires a detailed understanding of the interpreter, since its automatic nature makes it inherently different from a human equivalent. Educational programming environments often try to make the mapping more explicit with some visualization of the ongoing process: the trace left by the LOGO turtle, or some other exposition of the changing state of the interpreter.

« 14 » This seems to contradict a famous piece of advice coming from no less than E. W. Dijkstra. Speaking of anthropomorphism in computer science, he noted:

"The trouble with the metaphor is, firstly, that it invites you to identify yourself with the computational processes going on in system components and, secondly, that we see ourselves as existing in time. Consequently the use of the metaphor forces one to what we call 'operational reasoning,' that is reasoning in terms of the computational processes that could take place. From a methodological point of view this is a well-identified and well-documented mistake: it induces a combinatorial explosion of the number of cases to consider and designs thus conceived are as a result full of bugs." (Dijkstra 1985: 5)

« 15 » I agree with Dijkstra, the operational reasoning is an approach to overcome to be able to deal with the intricacies of computer science problems. The *reasoning in terms of the computational processes*, however, is not only a step in growing a more mature understanding of complex systems, but also something we need for the "thinking about thinking" that opens up the epistemological value of programming.

Conclusion

« 16 » In his abstract, Goldenberg claims that

"Formal educational systems set standards and structures to ensure some common learning and some equity across students. For a curriculum to tap curiosity and the drive for challenge, it needs both the playful looseness that invites exploration and the structure that organizes contents.”

I agree that the approach of blending puzzles with more structured content has great potential, but finding puzzles apt to educational goals is hard and expensive, especially if one wants to cover a significant part of the school curriculum.

« 17 » I am also convinced that programming can be a key language “in children’s mathematical learning and creativity” (§51), but for this we need to be careful not to restrict unnecessarily the expressivity of the programming tools we give to pupils. A more general emphasis on informatics (rather than programming) as the science of precise descriptions for information processing could open more powerful ideas in the minds of students.

References

Dagienė V., Sentance S. & Stupurienė G. (2017) Developing a two-dimensional categorization system for educational tasks in informatics. Informatica 28(1): 23–44.

Dijkstra E. W. (1985) On anthropomorphism in science. EWD936. E. W. Dijkstra Archive, Center for American History, University of Texas at Austin. https://www.cs.utexas.edu/users/EWD/ewd09xx/EWD936.PDF

Lewis C., Esper S., Bhattacharyya V., Fa-Kaji N., Dominguez N. & Schlesinger A. (2014) Children’s perceptions of what counts as a programming language. Journal of Computing Sciences in Colleges 29(4): 123–133.

Lonati V., Malchiodi D., Monga M. & Morpurgo A. (2017) Bebras as a teaching resource: Classifying the tasks corpus using computational thinking skills. In: Proceedings of the 2017 ACM conference on innovation and technology in computer science education. ACM, New York: 366–366).

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

Mattia Monga is an Associate Professor at Università degli Studi di Milano (Department of Computer Science). His research interests are mainly in the field of software engineering, system security, and computer science education. Since he believes it is urgent to change the common misconception of informatics as the mere use of information technologies, he founded, together with Carlo Bellettini, Violetta Lonati, Dario Malchiodi, and Anna Morpurgo, the working group ALaDDIn to spread informatics as a science among the general public (https: //aladdin.unimi.it/). It is also the National Bebras Organizer for Italy.

Received: 27 May 2019
Accepted: 12 June 2019

Author’s Response

Constructionist Curriculum Construction, Nutritional Supplements, and Language

E. Paul Goldenberg
Education Development Center (EDC), USA
pgoldenberg/at/edc.org

> Abstract • Crafting constructionist supplements to enrich curriculum is not easy; crafting a full set of constructionist-designed materials for day-to-day use by students and teachers is downright hard; both are possible. If one chooses to build in programming, decisions about what computer language has the “ideal” characteristics may depend on the specific subject matter or purpose to which that language will be applied. Mathematics, even for young children, imposes demands on that programming language – among them, the ability to create and compose functions – that other expressive purposes may not.

« 1 » I was glad to be reminded of the quotation from Neil Postman and Charles Weingartner with which Evgenia Sendova and Pavel Boytchev begin their commentary: the aim to keep children as question marks perfectly captures the spirit of the 1964 Wirtz materials on which I spend over 25% of my target article and more here. Also, I became aware that my parochial use of the word “curriculum” is potentially misleading. In the US, the word typically refers to the classroom text materials that a school adopts for use. I easily forget that elsewhere, “curriculum” means “syllabus,” only the guidelines for which publishers or teachers must then create classroom materials. I apologize for any confusion my ambiguity may have caused. To clarify, the item I call “Wirtz curriculum” is a comprehensive text series for elementary school mathematics: student books and practice materials for grades 1–6 and teacher guides for K-6 (see my §12). Throughout my article and this response, please interpret my use of “curriculum” to mean students’ and teachers’ *classroom materials*.

« 2 » While no part of my purpose in the target article is to promote an out-of-print 17-volume school text from 1964 that is not even of my own creation, or to promote a particular programming language that is also not directly influenced by me, I devote even more time to them here in this response, because understanding *them* better may help clarify the two main purposes of my article:

a to show that it is possible to develop a “conventional” print school-text that is scalable and effective (in schools, usable by teachers, published by a large commercial publisher) and yet teaches *mathematics*, not just arithmetic, by teaching mathematics *through* arithmetic and by letting students *do and create mathematics*;

b to show how, though programming is *not* an essential component of constructionist thinking, an appropriately designed programming language, learned and used thoughtfully, can provide even young children with a valuable expressive and exploratory medium *for mathematics* and can in that way support, deepen, and enrich the learning of that mathematics.

My focus is mathematics, not informatics.

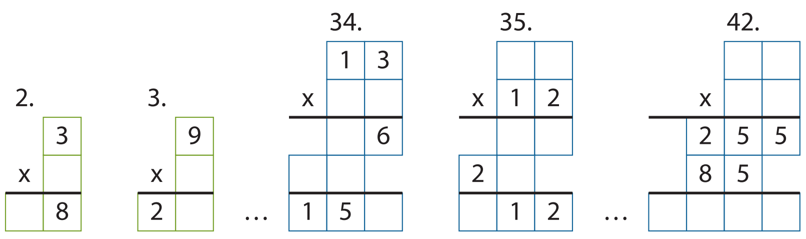


Figure 1 • Learning the multiplication algorithm inside out and backwards (from Wirtz et al. 1964a: 43).

Curriculum design requires theory, craft, and anxiety-tolerance

« 3 » That constructionism does not require programming is well known: In describing the new constructionism he presented, Seymour Papert himself often used non-programming examples, including his boyhood interest in gears and the samba school. Yet, understandably, Papert put great stock in programming, as does much discussion on constructionism. As Chronis Kynigos and Gerald Futschek say (2015: 281), "bricolage with expressive digital media has a primary role" in constructionist learning.

« 4 » Both commentaries – the one by Sendova & Boytchev, and the one by Mattia Monga – give great non-programming examples of opportunities for children to think and create in specifically mathematical contexts. Kangourou is an excellent example of stimulating mathematical problems, not dependent on computers or projects; Bebras connected with computational thinking is also not reliant on a machine. Both are widely known and prized. Sendova & Boytchev (§10) value not only children inventing puzzles, but getting teachers, educators and designers to value it as well. Clearly, I concur – this is my own experience and consistent with claims I make in my article – but that raises a long-standing anxiety: I wish I had more trustable evidence to support my claims, and confess that I do not see how to get it. Quoting Richard Noss and James Clayson,

"instructional effectiveness depends on many variables, not least the nature of technology, a field that is chaotic in the literal sense: tiny changes in, for example, the user interface can make massive changes in learning. The primary point is that in order to 'test' theory, it is necessary to maintain a gap between the pedagogical strategies at stake and the theories that motivate them […]" (Noss & Clayson 2015: 286)

« 5 » Even print materials have a "user interface." The amount, nature, and clarity of language can make a huge difference, especially for young children. Even tiny visual details can matter. When we were designing *Think Math* (EDC 2008) inspired by Robert Wirtz et al.'s *Math Workshop* from 1964, we first used the original 1964 form – lacking white spaces between columns – of the puzzle shown in Figure 9 of my article. By focusing on students' *experience*, not just on overall outcomes, we saw many students looking for patterns along horizontal rows and, finding none, asking for help. Though help was easy to give, and sufficed, students' *experience*, if only briefly, was the I-do-not-get-math message that nags at some children until they believe it. Our refinement changed nothing but layout, adding space between the columns. Students' mathematical learning seemed roughly the same both ways, but that extra space between columns made an observable difference in students' independence and in the smoothness of their learning.

« 6 » Alas, there are many such tiny things, so many that we cannot possibly notice them all or convey them reliably to the next generation of designers so that the field grows the way we imagine automotive engineering to grow. Any intervention – curriculum, teaching strategy, whatever – is a complex object, a mélange of so many craft elements along with the undergirding theory that even if it "works," it is hard to be sure why, or know what seemingly irrelevant tweak could have made it work better, or worse. What element is most responsible for success? How do the features interact? How much of the intervention's effect on students is attributable to its effect on the *teacher*? The impracticality of teasing out how such intertwined variables affect the outcome leaves us to "maintain a gap" and, at least for the moment, tolerate our ignorance. Curriculum design needs theory, art, design, and craft, but cannot quite achieve engineering.

The role of a vision of the discipline

« 7 » In §6, Monga says, "writing systematic textbooks with a large collection of repetitive exercises is easier than designing engaging activities for a whole curriculum." That is absolutely correct and is, to me, what made the Wirtz curriculum – a paper-and-pencil product published over a half-century ago – so incredibly remarkable. As I see it, what made that possible was a particular, and consistent, *vision* of mathematics (thinking, not just knowledge) and of curriculum. Without computers and with limited access to manipulatives – its teacher guide explains how (and why) to build a geoboard, not yet commercially available – it managed to make every student page (!) serve both practice and intellectual growth, with some element of curiosity-building surprise and cause to *think*. It was "conventional" in the sense of covering "standard" stuff, having a major publisher, and being widely used, even republished by the state of California after Britannica, its original publisher, stopped. Yet its structure and style were anything but conventional or standard. For one thing, it beautifully finessed the concept/skill battles. *Concepts* are constructed by abstracting commonality from variety; *skills* (whether in arithmetic, soccer, or piano) are built through (largely repetitive) practice. The Wirtz curriculum cleverly combined both. Consider just this one example of teaching a standard algorithm for multi-digit multiplication. On a page of forty-four problems (five from that page are shown in Figure 1) the accompanying text was spare:

"Mother liked to make puzzles out of practice examples. Her rules were simple: Complete each example by writing the correct digit in each empty box. Write nothing outside the boxes." (Wirtz et al. 1964a: 43)

« 8 » The first few puzzles are one-digit by one-digit, each solvable just by knowing a multiplication fact, but look how different the mathematical *thinking* is – the mental search for the right fact – for puzzles 2 and 3. Over the course of the page, students get a *lot* of practice with low-level skills (perhaps rehearsing many facts before finding the ones they need) but each puzzle also engages high-level thinking, touching number theory, estimation, structure within the distributive property, and more. Students are learning the multiplication algorithm inside out and backwards. Literally. Like a great composer's études – focused practice, but also genuine music – this focused practice is genuine mathematics. The teacher guide also generalizes the idea.

"'Gremlins' are allies of arithmetic teachers. After school, they come in and erase part of the arithmetic work done on the chalkboard during the day. They do a most selective job of erasing. In the morning, when curiosity over the gremlins' work has reached a healthy pitch, the teacher asks if the pupils can reconstruct what the gremlins erased." (Wirtz et al. 1964b: 43)

« 9 » It then extends the idea by showing how much challenge *and practice* students get when they invent their own such puzzles with unique solutions – first performing a standard multi-digit multiplication (practice!) and then mimicking the gremlins and facing challenges like How many digits can I erase? and Which ones?

Curriculum vs. supplement

« 10 » Why do I spend well over 25% of the target article (and even more space here) on a paper-and-pencil *school* curriculum, when constructionism is so closely associated with Papert, computers, independent projects, and informal or out-of-school learning?

« 11 » To me, the crux is that while materials like Kangourou, the many clever tools that Sendova & Boytchev describe, and the EDC microworlds that my colleagues and I are designing (§§57–77) can be powerful resources to support a high-quality curriculum, they are not, themselves, curriculum materials. Scalability requires more. What the Wirtz texts show is that it is possible to build a *formal* school text – not supplementary activities or after-school classes or maker spaces, all unquestionably valuable but optional – that is both mathematically deep and still usable ("scalable") in schools. For constructionism to effect change broadly in education, we must somehow be able to use its design ideas – ones that build students' agency and let them explore and create while learning a heritage of ideas and knowledge – to accommodate a scalable/usable process.

« 12 » Framework-specification of curricula makes it even harder to create materials that let children be intellectually creative, even with computers and especially with only paper as a medium. As Monga says (§5), that is hard to achieve even for a group that has the time to collaborate, think, reflect, test, and revise – luxuries not available to classroom teachers. It may be the ambiguity in my use of "curriculum" that led Monga then to state "I doubt a single teacher […] could sustain the necessary creativity […]" Of course! No teacher *should* have to figure out how to find that time and sustain that creativity. That is the very purpose of curriculum materials (and supplements). It is unreasonable to imagine that teachers would also be able to invent a sensible set of materials from scratch (required coverage, content accuracy, coherent order, clear focus, effective activity, sufficient practice, appealing craft, clear writing and graphics), given a list of objectives, for even one subject, let alone for all of their subjects. (Astonishingly, at least in the United States, many districts are now expecting teachers to do exactly that!)

« 13 » The Wirtz curriculum illustrates something else. Monga (§6) writes "we all 'learned' some things by a rote or strictly algorithmic approach […]." I assume that is connected with my statement (§11), "Solving a puzzle is different from working an exercise: the process is not rote or algorithmic […]." Monga is correct and could make an even stronger claim: Some things, like people's names, are learned *only* by memory; they cannot be reasoned out, discovered, or learned "creatively." But if his §6 is a response specific to my §11 – as if to say we all learned *some of our mathematics* that way – that raises two concerns for me: the "all" and the "mathematics." "We all" are the people who did thrive adequately in class. Others did not. To me, Monga's statement reflects the perspective of the already "arrived" university computer science professor and not the view of the path to arrival, seeing learning and teaching mathematics to young children from the perspective of classrooms and teachers and psychologists (and even many mathematicians).

« 14 » Being a mathematician – or, for that matter, a good detective, a good diagnostician of the ailments of a car or person, or a good historian or paleontologist – requires being able to figure out things that one has *not* already learned, by rote or in any other way, being able to select and apply heuristics, not just algorithms, and *invent* new heuristics and algorithms. If we do not devote *some* part of any curriculum to that kind of figuring out, we are leaving that important skill (yes, a *skill*) to chance, training children only on the rote and hoping some will figure out the rest on their own. We all did learn "some things by a rote or strictly algorithmic approach," but that adequately served only some of us, *not* all. If the curriculum designer's vision foregrounds mathematical thinking (see, e.g., Cuoco, Goldenberg & Mark 1996) as does Wirtz, we could *all* plausibly learn more. My intent in §11 was not to reject a method (rote), but to add one. Puzzles are puzzling precisely because they can *not* be solved just by routine; they can *incorporate* needed practice (the "rote" part) but are not *only* practice. That combination is what made the Wirtz curriculum so brilliant.

Optimizing a programming language for children doing mathematics

« 15 » I was initially surprised by the notion (Sendova & Boytchev's abstract) of programming being "'repurposed' to *wrap the math* in an attractive, yet educationally effective way." As I read on, I understood better. Though, in §54, I attached the idea of using programming as a language for learning and expressing mathematics to Sendova, I never mentioned that "my" thinking about programming that way derived directly from what I saw when I visited her and the Bulgarian Academy of Sciences in 1989 to see her work. (This view of programming was, as I coded it in my head, *their* idea, which we did say in our proposal to the NSF.) At first, the notion of "wrapping the math" clashed in my head with the notion of "expressing mathematics."

« 16 » But when I understood Boytchev's metaphor, it all became clearer. The many-thousand-years-old evolution of mathematical thinking involved abstracting structure and pattern from observations and experiments and later using the abstractions themselves as data for new knowledge and thinking. Classroom learning cannot start from scratch and rebuild the entire structure so, whatever else we do, we also depend heavily on knowledge captured in words and symbols, "the way knowledge is developed and transmitted between generations," as Monga put it (§11). The kernel of mathematics *is* wrapped in various languages – always one's natural language and mathematical notation (my §53), and other abstractions (e.g., graphs, geometric diagrams), all with subtleties that clarify some meanings but obscure others. To get to the mathematics through programming necessarily involves unwrapping it, like making the programming somehow transparent. My (§56) "expressiveness of a 'live' language," meant notation on a computer (in, say, Snap*!*) which can be run, giving us feedback on what it says. This is like how natural language is learned and how learning occurs through using natural language: we dialogue; we say something, and someone reacts, so we get to see what effect we have created. By contrast, notation on paper in conventional algebraic form just sits there, correct or incorrect, and gives no feedback. Without "rerunning" the code mentally, we may not find out what those symbols said.

« 17 » The computer language one chooses can certainly matter. In our proposal to the NSF to treat programming as a language for young children to express and explore their growing mathematical ideas, we proposed to use Scratch, familiar in increasingly many elementary schools and massively successful in attracting children to programming. As we worked, it appeared that Scratch would not adequately serve *our* purposes – children's *mathematics*, not introduction to coding – which is why we switched to Snap*!*. The two languages look very much the same, so teachers (and children) with prior experience in Scratch would have no trouble switching over, but Snap*!* lets students build their own mathematical functions – code that takes an input and returns an output – and compose those to build yet others, not currently possible in Scratch. In Snap*!*, functions are first-class objects, so children can map a function they create over a list. Some of what we are already doing with second-graders (e.g., providing the *combine steps* block shown in Figure 22 of my article, or enabling them to create blocks like the pink +2 block shown in §70) are not possible in Scratch.

« 18 » This is neither promotion of Snap*!* nor criticism of Scratch – they are optimized for different purposes, which is one reason why there *are* different computer languages. The C and Lisp families are optimized for different purposes; both remain essential. Nor is this a treatise on comparative computer languages expounding on Snap*!*'s features. However, knowing what a language *is* matters when evaluating how it is *used*. I agree with Monga that "If programming is supposed to be a key mental tool to express mathematical ideas […]," then "a programming tool with a limited potential of expression, based only on simple sequences of predefined steps [is] too limited even at [second grade]" (Restructured from §7). Yet expressiveness and form are not the same: the misperception that blocks-based means unsophisticated is so widespread that I feel compelled to address it.

« 19 » Differences in computer languages almost certainly do affect the metaphors and models that students build in their heads – instrumental genesis – and it might well be that first languages have a particularly strong influence. Therefore, as Monga says in §17, we should "be careful not to restrict unnecessarily the expressivity of the programming tools we give to pupils." Working in a low-level language, one that is closer to the hardware and operating system architecture (e.g., C++), draws attention to different things than does working in a high-level language (e.g., Logo). They satisfy different needs and each is optimized for its purpose. Both are text. This issue is not about blocks vs. text; form is not function.

« 20 » I may be misunderstanding Monga's intent in §§10f, but he seems to say that the metaphor of building blocks is somehow less linguistic, "screen[ing] off the epistemic power of language." I see it differently. The atoms of a *language* are, more or less (depending somewhat on the language), its *words*. The atoms of a *writing system* may be words, too – logograms as in Han Chinese characters or a block-based language – or may be tinier elements, syllabic or alphabetic, from which the written words are constructed as molecules. None of these variations bear on sophistication. When university students take computer science, they will need to know that computer languages, like natural languages, can differ not just in abstraction level and in vocabulary and local syntactic details, but in structure and metaphor (think imperative vs. functional or compare Scheme and Prolog); graduate students in linguistics similarly learn that natural languages differ not just in low-level detail but in surprisingly different syntactical constructs and semantic spaces.

« 21 » A block language, like a special-purpose text-based language, *can* be very limited, but it does not have to be. Vocabulary size and vocabulary power are independent attributes, and independent from the form in which the vocabulary is presented. Monga is right (§11) that "thoughts made by words and rich grammatical structures are closer to the way knowledge is developed and transmitted between generations." But words do not imply alphabet: think Han characters or a sign language or, for that matter, speech. Children need words and rich structures that admit to extensibility and offer powerful metaphors. And, as children develop more complex ideas to express, they need more vocabulary and richer structure. Just as learning one's native language begins simply, a mathematically expressive language for children can begin simply, as long as it has both fidelity to mathematics and the extensibility to serve well as the child's ideas grow. Nobody should be distracted by semicolons in writing English or computer code until (if ever) they are needed. And educators should not be distracted by whether the tokens of the language – the words that convey meaning – are typed or dragged.

« 22 » Giving children limited tools will restrict what they can express and, potentially, narrow how they learn to think. Giving them distracting or cumbersome tools is also restrictive. The optimal solution would seem to be some flavor of high-level language with a robust starting vocabulary, flexible (and extensible) data structures, functional programming capability, recursion, higher-order functions, low syntactic distraction, and a way to tailor the environment so students encounter new power only as they need and can use it, without having to sift past it or learn everything before they start. That could certainly be a text-based language like Logo, but typing is hard work and prone to error. Why not provide the same extensibility as Logo (or Python or JavaScript or any good text-based language) but present the primitive vocabulary of that language in draggable blocks that can flexibly create the full range of new vocabulary and structure?

References

Cuoco A., Goldenberg E. P. & Mark J. (1996) Habits of mind: An organizing principle for mathematics curriculum. Journal of Mathematical Behavior 15(4): 375–402.

EDC: Education Development Center (2008) Think math! Harcourt, Orlando FL.

Kynigos C. & Futschek G. (2015) Re-situating constructionism. Constructivist Foundations 10(3): 281–284. ▸ https://constructivist.info/10/3/281

Noss R. & Clayson J. (2015) Reconstructing constructionism. Constructivist Foundations 10(3): 285–288. ▸ https://constructivist.info/10/3/285

Wirtz R., Botel M., Beberman M. & Sawyer W. W. (1964a) Math workshop: Level D student book. Encyclopaedia Britannica Press, Chicago IL.

Wirtz R., Botel M., Beberman M. & Sawyer W. W. (1964b) Math workshop: Teacher guide, level D. Encyclopaedia Britannica Press, Chicago IL.

Received: 29 June 2019
Accepted: 4 July 2019