Constructionism

2018, Vilnius

Constructionism, Computational Thinking and Educational Innovation

August 20-25, Vilnius, Lithunia

Conference Proceedings

Edited by
Valentina Dagienė and **Eglė Jasutė**

Constructionism 2018 Conference

Constructionism 2018

Constructionism, Computational Thinking and Educational Innovation: conference proceedings

Organizers:

Vilnius University
Faculty of Philosophy and
Institute of Data Science and Digital Technologies
in cooperation with Lithuanian Computer Society

August 20-25, Vilnius, Lithuania

Edited by
Valentina Dagienė and Eglė Jasutė

ISBN 978-609-95760-1-5

The conference is partially supported by Research Council of Lithuania

CONFERENCE VENUE

Faculty of Philosophy
Vilnius University
Universiteto str. 9
01513 Vilnius, Lithuania
Located at the Old Campus of Vilnius University

The conference website: http://www.constructionism2018.fsf.vu.lt

Book of abstracts is available at: http://www.constructionism2018.fsf.vu.lt/book-of-abstracts

Full papers are available at: http://www.constructionism2018.fsf.vu.lt/proceedings

CONSTRUCTIONISM 2018 COMMITTEES

Conference Chairs

Valentina Dagienė, Vilnius University Institute of Data Science and Digital Technologies, Lithuania
Arūnas Poviliūnas, Vilnius University Faculty of Philosophy, Lithuania
Arnan (Roger) Sipitakiat, Chiang Mai University, Thailand

Scientific Committee / Reviewers

Tim Bell, University of Canterbury, New Zealand
Paulo Blikstein, Stanford University, USA
Pavel Boytchev, Sofia Universoty, Portugal
James Clayson, American University of Paris, France
Secundino Correia, Imagina, Coimbra, Portugal
Barbara Demo, University of Torino, Italy
Vladimiras Dolgopolovas, Vilnius University, Lithuania
Michael Eisenberg, University of Washington, USA
Gerald Futschek, Vienna University of Technology, Austria
Carina Girvan, Cardiff University, Cardiff, Wales,
Paul Goldenberg, Education Development Center, USA
Brian Harvey, University of California, USA
Bulgaria
Arthur Hjorth, Northwestern University, USA
Celia Hoyles, University College London Institute of Education, UK
Eglė Jasutė, Vilnius University, Lithuania
Tatjana Jevsikova, Vilnius University, Lithuania
Anita Juškevičienė, Vilnius University, Lithuania
Ivan Kalaš, Comenius University, Slovakia
Eric Klopfer, Massachusetts Institute of Technology, USA
Witold Kranas, OEIiZK, Poland
Chronis Kynigos, National and Kapodistrian University of Athens, Greece
Manolis Mavrikis, University of London, UK
Mattia Monga, Università degli Studi di Milano, Italy
Jens Mönig, SAP, Germany
Erich Neuwirth, University of Vienna, Austria
Richard Noss, University College London Institute of Education, UK
Mareen Przybylla, University of Potsdam, University of Potsdam, Germany
Mitchel Resnick, MIT Media Lab, Massachusetts Institute of Technology, USA
Ralf Romeike, Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany
Ana Isabel Sacristán, Cinvestav, Mexico
Jenny Sendova, Institute of Mathematics, Bulgaria
Giovanni Serafini, ETH Zurich, Switzerland, Italy
Wolfgang Slany, Graz University of Technology Institute of Software Technology, Austria
Gary Stager, Constructing Modern Knowledge, Torrance, USA
Eliza Stefanova, Sofia University "St. Kliment Ohridski", Bulgaria
Gabrielė Stupurienė, Vilnius University, Lithuania
Maciej Syslo, University of Wroclaw, Poland
Márta Turcsányi-Szabó, Eötvös Loránd University, Hungary
José Armando Valente, State University of Campinas, UNICAMP, Brazil
Jiří Vaníček, University of South Bohemia, Czechia
Michael Weigend, University of Münster, Germany
Uri Wilensky, Northwestern University, Evanston, USA
Michelle Wilkerson, University of California, USA

Local Organizing Committee

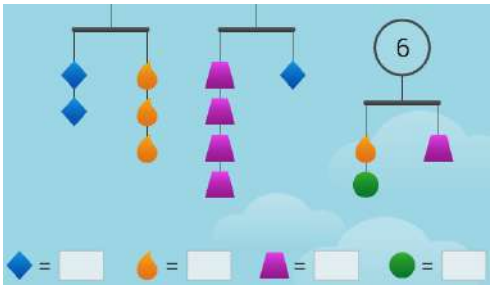
Vida Jakutienė
Aldona Mačiūnienė
Saulius Maskeliūnas
Olga Suprun

Teaching Children to be Problem Posers and Puzzle Creators in Mathematics[1]

Paul Goldenberg, *pgoldenberg@edc.org*
Education Development Center (EDC), Waltham, MA, USA

Abstract

Seymour Papert's 1972 paper "Teaching Children to be Mathematicians Versus Teaching About Mathematics" started with the summary statement "The important difference between the work of a child in an elementary mathematics class and that of a mathematician is not in the subject matter…but in the fact that the mathematician is creatively engaged…." Along with "creative," a key term Papert kept using is *project* rather than the common notion of *problem*. A project is not simply a very large problem. It centrally includes a focus on sustained and active engagement. The projects in his illustrations were essentially *research* projects, not just multi-step, fully-prescribed, build-a-thing tasks, no matter how nice the end product might be. A *mathematical* playground with enough attractive destinations in it draws children naturally to pose their own tasks and projects—as they universally do in their other personal and group playgrounds—and to learn to act and think like mathematicians. They even acquire conventionally taught content through that play. Physical construction was always available, and appealed to such thinkers as Dewey, but for Papert computer programming, newly available to school, suggested a more flexible medium and a model for an ideal playground.



Some cells are already filled. Fill in the rest.

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

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

Figure 1. Two puzzles that introduce algebra's logic and then its notation. Children can invent their own.

A fact about playgrounds is that children *choose* challenge. In working and playing with children I've seen that puzzles tap some of the same personally chosen challenge that a programming-centric playground offers. Children are naturally drawn to intellectual challenges of riddles and puzzles (ones they learn and ones they invent); and adults are so lured by puzzles that even supermarkets sell books of them. So how do real puzzles and school problems differ? What's useful about *creating* a puzzle or posing a problem? How might puzzles and problem posing support mathematical learning? And what's constructionist about this? This plenary will try to respond to these questions, invite some of your own responses, let you solve and create some puzzles, and explore how problem posing in programming and puzzling can support mathematics even in an age of rigid content constraints.

Keywords

Problem posing; puzzles; mathematics; algebra

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

Teaching children to be problem posers and puzzle creators in mathematics

Papert's early work and the origin of constructionism largely were 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?" I want to share some ideas that, to me, exhibit the essential elements of constructionism and could easily be core to even moderately conservative school practice.

I, too, love playing with kids outside the classroom. There's more freedom and it's easier. But we all know that if we really want to touch many children's lives, we need to find a way to find them where they are. They are in school. I think it's possible.

Children choose challenge

Not all children; not all the time; but mostly. 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 the 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.

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, kids 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.

Children also put effort into figuring out how things work. Schulz & Bonawitz (2007) showed pre-schoolers 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's tempting to relate the first group's experience to the situation children often experience in school mathematics, where common pedagogy (at least in the U.S. 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.

Kids also love riddles, challenges to logic, interpretation or perception. And just as they spontaneously add challenge to their playground activities or jigsaw puzzles, they'll add to the repertoire of riddles 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.

The point is the challenge. When it's not there, children are bored. When they're bored, they *invent* challenge. In school, that inventiveness can be to the dismay of their teachers, whose response may dismay the children, but that won't stop the drive for the challenge.

Why puzzles?

Kittens stalk and pounce to make their hunting skills sharp and they scratch to keep their claws sharp; that's 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 and tropical jungle. Keeping our *minds* sharp is what makes our species successful.

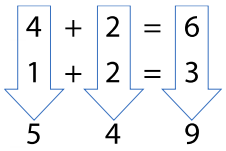
That has implications for learning. In *our* species, it is adaptive for the young to be a bit distractible and *not* to focus too narrowly. Because we live in such varied environments we cannot have "built in knowledge" about which features will matter most for survival. As children, we watch social behaviour (whom should we copy, whom should we stay close to, whom should we stay away from?), animals (food or danger?), artefacts (how do they work?), math lessons (who knows?, maybe they're important) and everything else.

Some distractibility and lack of focus are assets to a child, but less so for adults who must earn their living, whether by blow-darting the rabbit (while avoiding the tiger) or by generating research papers or by teaching children. But adults still have to keep their minds sharp. Adults argue the merits of ideas—politics, religion, co-workers—even when the *practical* value of the argument is near zero. It's a mental exercise. Puzzle books 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.

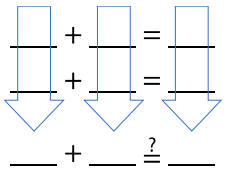
Puzzles and surprise in mathematics learning

In 1964, 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, et al., 2008). 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.

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 + 1 = 5 or 1 + 1 = 2, and the teacher would write it at the board. Another child is asked to suggest a new equation, which the teacher carefully lines up directly underneath the first. Then the teacher has the children add vertically, displaying the results like this.

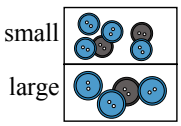


Do these three new numbers make a true addition equation? The teacher completes the bottom row of numbers to read 5 + 4 $\stackrel{?}{=}$ 9. Surprise! Will this *always* happen, or did they just get lucky? Children are given the challenge of finding a pair of addition sentences that *don't* work. Seven-year-olds are sure they can find some and set off busily, getting lots of practice.

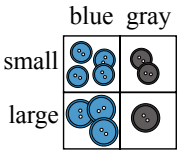


Of course, they *will* find some (they think) and report them excitedly, but the preponderance of cases that do work will get even seven-year-olds to doubt the counterexamples and check to see if they've made a mistake. This research is hardly a project in Papert's sense. The problem may not even last more than one classroom period or so. But it *does* generate curiosity, the creative engagement that Papert referred to as the experience of the mathematician.

Their research convinces them of a result, but if we don't 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, colour 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

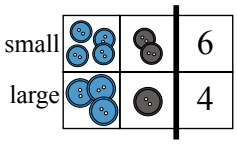


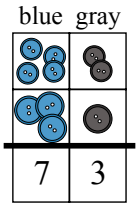
large grey button" and "how many small blue buttons do you have?" After sorting by a single attribute, they can learn to sort by two attributes.



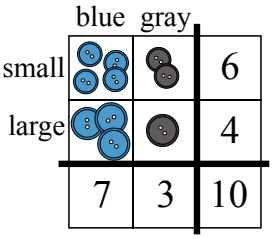
Now, when we ask how many small buttons and how many large, we are summarizing the rows, and we can write that summary.

We can similarly summarize the number of buttons by colour (columns).





Once children really have 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's all the buttons. Second grade students comfortably replace buttons with numbers and then use that structure as part of their reasoning.



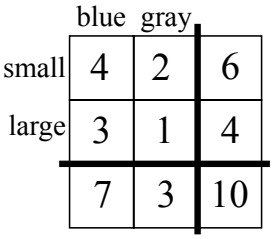
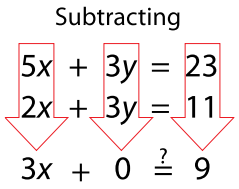


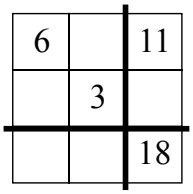
Figure 2a, 2b. Actual buttons replaced by the number *of buttons.*

Reading across, children see 4 + 2 = 6 and 3 + 1 = 4; adding down the columns, they get 7, 3, and 10, which *must* make a true addition statement.

Subtracting down isn't always possible for 7 year olds—depending on the situation, it might require negative numbers, and the meaning changes, too (it doesn't 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 Fig. 2b), the arithmetic still works and produces a true addition statement. Subtracting to see how many more small buttons than large, we get 1 = 2 – 1, and that exact same logic will be essential in algebra a few years later!



The format isn't just a trick, or a school artefact; it's the structure of any spreadsheet that subtotals the columns and rows and has a grand total. Wirtz, et al. (1964) used this format as a puzzle (right), and as a route into multi-digit addition and subtraction (fig. 3).



Which cell might you fill in first?

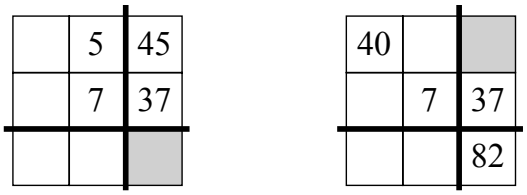


Figure 3. The same arithmetic presented in 3a as an addition puzzle 45 + 37, with the grey square as the sum, and in 3b as a subtraction puzzle 82 – 37, with the grey square as the difference.

For 9- or 10-year olds, this structure also models the multiplication algorithm. Instead of colour 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 (fig. 4a).

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 don't want to count! Instead, we use an abstraction (fig. 4b), 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. In fact, 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.

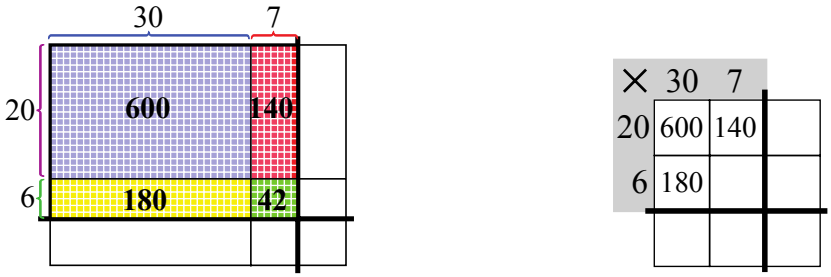


Figure 4. An array model of multiplication true to scale (left, figure 4a) and abstracted (right, figure 4b).

Sawyer suggested other ways that even very 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 5, for example, shows what a standard worksheet might present as 16 unrelated addition/subtraction practice exercises for 7-year olds, but structured in a way adds a bit of intellectual challenge—how-do-I-do-this?—and foreshadows systems of equations that the children will meet several years later.

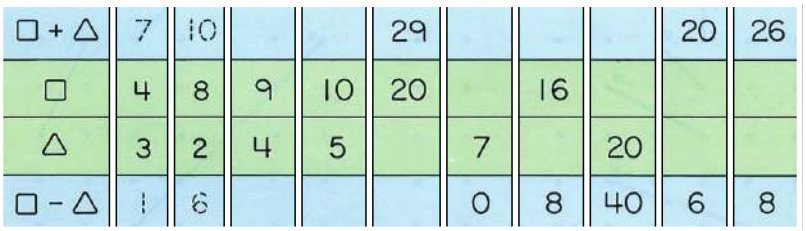


Figure 5. A practice exercise for 2nd grade, foreshadowing systems of equations. (Wirtz, et al., 1964)

Again, it's 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.

One of the most powerful introductions to algebra that I've seen is Think-of-a-number tricks, also from Wirtz, et al. (1964): 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!!!

For 9- or 10-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 whatever, tucked in a bag or bucket where we can't see them—only the secret keeper knows the number inside. When I give the instruction "add 3," I *know* about *those* grapes, 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 grapes in it doesn't matter! There's one grape left, and we can *see* it!

Even after the usual huge smile and the cry "I *get* it!!," seeing it once isn't 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.

They also need chances to study the trick inside out and backwards, 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 grapes, so ten grapes must be hidden in the two bags. Suri's secret number—the grapes in *one* bag—must have been 5.

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.	*4*			*0*
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 6. Using bags and grapes to introduce 3rd graders to algebraic notation and solving equations. (Problem from Wirtz, et al., 1964, reworked for 3rd grade based on Mark, et al., 2014.)

I've recently been introducing a new crop of 8- and 9-year olds to algebra this way and told them that they'd 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 about *the* mathematics of this trick but about mathematics, itself. She could just make up a rule, *any* rule, and it was then up to her to figure out its implications. That is so like watching a child program, see the effect, decide whether that effect is desired or not, and then decide what to do next.

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.

These are five to fifteen 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: 2 *b* + 6. No discussion of variables; no explaining about letters standing for numbers; 2 *b* + 6 is brief, but 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.

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'm sure, is what Schulz & Bonawitz (2007) saw: children play longer and more curiously when there's something they don't understand *and they believe that they can figure it out*.[3]

This was not a classroom assignment. The children didn't have to do this and wouldn't 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 couldn't 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?

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

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's 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's very possible to learn mathematical *content* that way, and some people like that order and simplicity. But mathematical *thinking* can't 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 can't 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're good ones, especially for younger students.

[2] This imagery doesn't represent "divide by 2" well unless the numbers of bags and marbles are both even. The imagery is adaptable to "negative grapes," but frankly awkward. So we need to be clear that the imagery is not the goal, not a "new method" for algebra. But it's 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 don't put effort into study that *we* believe would make them better. But *they* don't 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.

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]

That challenge, and also the act of being a creator, may be part of *why* construction of a sharable puzzle appeals to kids, but the appeal is yet another reason to have kids create.

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.

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 7, and even simpler ones for real beginners.

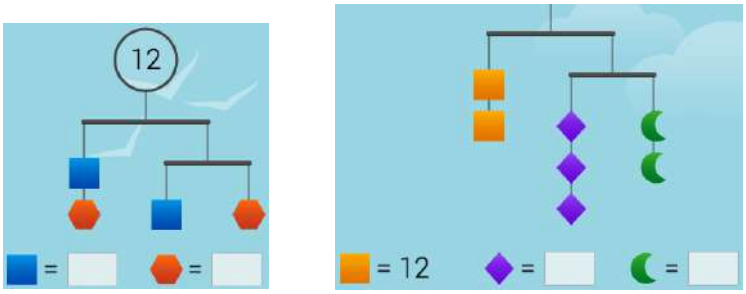
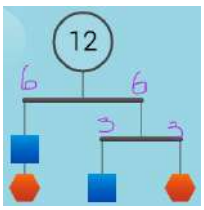


Figure 7. Two relatively simple mobile puzzles.

The mobile's total weight might be given (above, left) and players must figure out how much the blue red objects must weigh in order for this mobile to balance. Or (above, 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.



Players often just work these out in their heads, but the app offers them other options: they can scrawl annotations on the screen, like this. →

They can also create equations by dragging off a copy of a horizontal beam $\square = \bigcirc$, or the entire mobile $12 = 2\bigcirc + 2\square$, and substitute these into other equations (or the mobile) to derive new information, like $12 = 4\bigcirc$. The app also lets them factor 2 out of equations like $12 = 2\bigcirc + 2\square$ to derive new equations $6 = \bigcirc + \square$ and to drag a common element out of both sides of an equation like $4\bigcirc = 2▼ + \bigcirc$—to see $4 \quad = 2▼ +$ —and to get $3\bigcirc = 2▼$. Otten, et al. (2017) describe how 11-year-olds used explicitly algebraic correct reasoning in the context of informal notation and manipulations of a physical hanging mobile.

The mobile puzzles are essentially systems of equations. Some students are intrigued by the fact 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 the fact that they can get equations mostly don't 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.

Some of the puzzles are quite challenging, like the ones in figure 8a, without being required to, students really persevere because they're sure they can solve the puzzles if they keep at it.

[4] This is especially true in creating a good MysteryGrid puzzle or Who Am I puzzle, not described here, but part of the SolveMe suite of puzzles mentioned below.

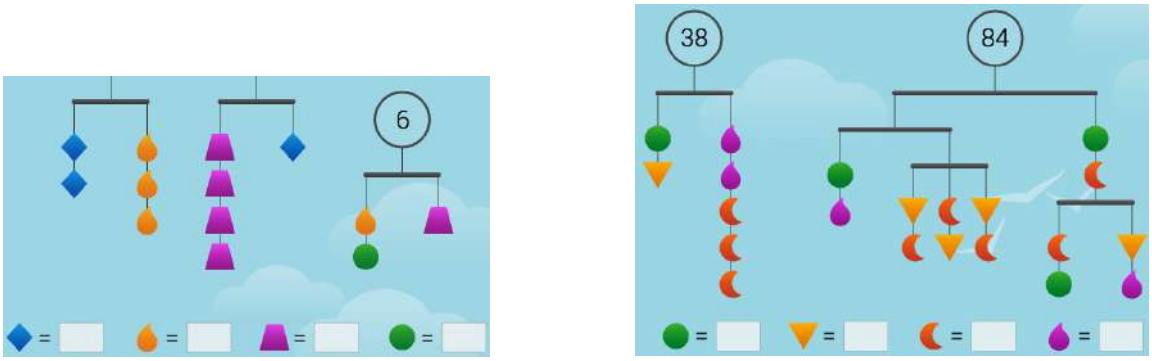


Figure 8a. Two mobile puzzles at a more advanced level.

As I'd 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 these.

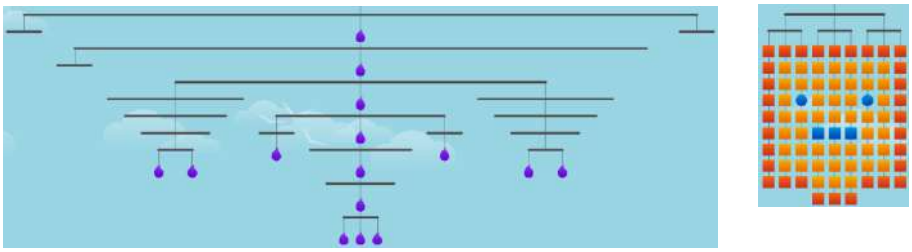


Figure 8b. Two mobile "puzzles" invented by users, apparently intended only as art.

Manousaridis (2018) regularly encourages students in grades 2 and 3 to create their own puzzles as posters after solving some on line. 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.

The 9-year-old who created the puzzles shown in figure 9 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."

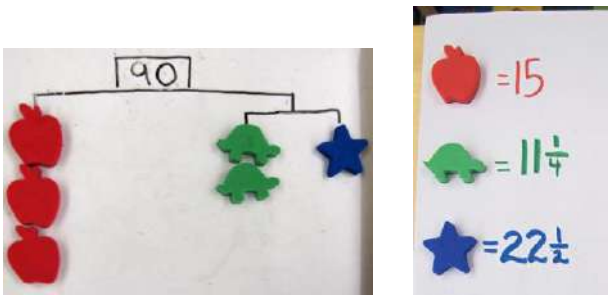


Figure 9. A mobile puzzle invented by a nine-year-old to challenge her classmates to use fractions.

Programming in mathematics

The examples and contexts described above have been very far from the programming-centric proposal that Papert made in (1972) but, as I've tried to show, well in line with the mathematical creativity, exploration, and research projects that he regarded as *doing* mathematics rather than learning about it—creating and solving one's own problems versus learning mathematical facts and solving problems

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 natural part of learning mathematics. For it to be "a natural part," it would need to be learned along with the mathematics, growing over time just as the mathematics does, and used in ways that *support* the mathematics and don't compete with it by seeming to be a separate venture—fun stuff but disconnected—or by creating excessive 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.

Noss & Hoyles (2018) 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.

Mathematics really 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. Unfortunately, if used inappropriately, both can also get in the way. In particular, mathematical notation is too often used as the entry point to new ideas rather than a concise way to record ideas that are already well understood. Consider that the third graders knew intuitively that doubling  produced  , and six-year-olds, when asked *verbally* (not in writing) what five eighths plus five eighths might be, are happy to respond "ten ayfs," then ask what an ayf is. They never answer ten sixteenths. The distributive property is *built in* to our logic early. But when the word "distributive property" is introduced in third grade, 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. Despite your mathematical literacy and knowledge, probably even you zipped past the string of symbols without reading closely enough to see if it was correctly typed. Processing such a string of symbols takes focus and effort, and therefore can't 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.

And neither natural language nor mathematical notation is particularly good at expressing process or algorithm. That's 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, a programming language is a notation that can be *run* and gives direct feedback on what it does.

ScratchMaths (Noss & Hoyles, 2018) is one beautiful example of infusing programming directly into grade-level-required mathematics for 9- to 11-year olds. At EDC, we are building a full-year mathematics+programming curriculum for 7- to 8-year olds as a stepping stone to doing the same for all of the first six grades of school, using the playful and puzzle-centric ideas of Sawyer (2003) and Wirtz, et al. (1964) as redeveloped in our *Think Math* curriculum (Goldenberg & Shteingold, 2007a and 2007b). First programming experiences for young children can be quite open—moving a robot around, or just code-streams of interesting effects—but if we are explicitly intending to show 7-year-olds how they can use programming as a language to experiment with and express the mathematics they are currently learning, the first coding experiences need to be rather simple while, at the same time, leaving plenty of room for puzzling and exploring.

Here is one environment EDC is currently exploring with children. The computer displays a number line, optionally settable for any range, with only one number labeled. The ticks just 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 youngest children, the ticks indicate consecutive integers, with only 0 labeled (and intentionally chosen not to be the leftmost mark on the line). The 7-year-olds are given a palette of tools from which they can draw. For example, they may have  ,  ,  ,  and, later, a **repeat** block.

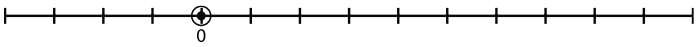


Figure 10. A simple number line with ticks representing consecutive integers.

Clicking a tool performs the indicated arithmetic, shows the corresponding movement on the line, and labels the result. For example, if the sprite is at 3, clicking the **+5** block moves the sprite 5 spaces right (arc optional) and marks 8 there (figure 11).

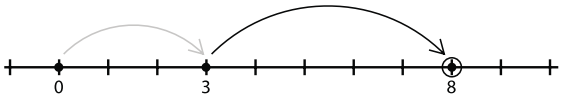


Figure 11. A move of  *from 3 to 8.*

Children explore the tools with very open puzzles like "Try to label all the numbers 0 to 10." When they have learned how blocks can be snapped together to create a script, they get more focused puzzles of increasing challenge that require experimenting, planning, mental arithmetic, and prediction of results. Two puzzles are shown in figure 12, as they might appear to children.

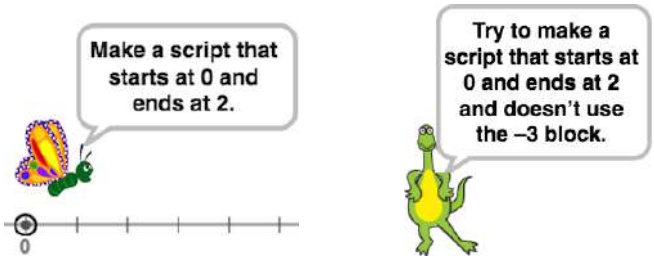
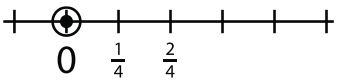


Figure 12. Two programming puzzles.

More challenging problems can follow: Create two different scripts that…. Children can also be asked in class discussion to analyze and explain. For example "explain why this script moves  from 0 to 1. If you start at 4, where will this script move you? Is there a way to move from 0 to 1 in exactly 2 moves? What's the *shortest* script that…?"

Because activities like this "have legs" mathematically, they can grow with the child and be used in later grades. At the simplest, the very same activity can be used on a "zoomed in" view of the number line, to explore fractions. The children's tools now include  , , etc.; similar tasks would challenge them to mark ¼, ²⁄₄, etc. Still other variants—like changing the tools to ±6 and ±9 and challenging children to label all the numbers they can—show how versatile this format is, capable of addressing later grade-level standards (e.g., factors, multiples, common factors, analyzing patterns, building fluency with multiplication facts) and foreshadowing in grade-appropriate ways ideas children will make explicit later. Other elementary school projects involve functions (e.g., ) and let students build and compose their own. And creating code that generates squares from repeated moves, rows from repeated squares, and arrays from repeated rows—and the similarities of the algorithms inside **draw row** and **draw array**—illustrates 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. This example also shows how the activity directly supports *mathematical* content mandated for schools. The **draw array** block depends on two parameters, the dimensions of the array. A **draw rectangle** block is a further (though simpler) abstraction, using the same two inputs but drawing only the border of the array. For either of these, 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.

And, 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 in the usual sense, but construction with the computer rather than just use of the computer to manipulate pre-designed 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

Secondary students, too, need places to be mathematically creative. The mission of Beauty and Joy of Computing (BJC, 2017) is broadening participation in computer science with a focus on letting students see not just the joy of creation and beauty in the objects they can produce through programming, but also beauty in the programs themselves. With this as one goal, it introduces the elegance of recursion and higher order functions. It manages to make these reputedly "difficult" topics accessible by virtue of the lucid visual imagery of the Snap*!* visual programming language, a block-based language that is reasonably characterized as essentially Scheme disguised as Scratch. Two BJC excerpts involving recursion were used 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 quite a long sentence. 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.

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 a math course, its activities naturally touch—and help teach—many conventional mathematical content topics, and our approach to programming is consistently focused on mathematical/computational thinking.

The point of introducing various contexts—the arts, linguistics, etc.—is partly to meet the varied interests of students, but much more to show how broadly *they* can allow themselves to wander, how much they can tailor their independent projects, for which even the AP framework allocates time, in their own personal direction.

Playgrounds

Giving even very young students a way to think algebraically using bags and grapes 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.

A question, based on wild final thoughts, pure opinion that I might disown tomorrow

A few states, including Massachusetts (the one that I live in) have begun to develop frameworks for computational thinking (CT) across the grades (DESE, 2016). CT is variously defined but always includes elements like abstraction, algorithm, modelling and simulation, programming, and data (with an implication, not reflected in all implementations, that "data" means *big* data). Not surprisingly, there has also been a proliferation of on-computer and "unplugged" activities, *not* involving programming, to help develop this thinking.[5] The difficulty of adding anything else to an already jam-packed school day has led to a lot of talk about *integrating* CT activities into the existing content areas, particularly science and mathematics (e.g., EDC STEM+CT, 2018), but also language. In my opinion, some of the suggestions are shallow, but that should be no surprise at a time when the whole effort is so new.[6] Still it got me to thinking about why my own inclination has been *toward* programming, not away, and toward abstraction, and algorithm rather than modelling and simulation, whenever the aim is explicitly to integrate with other subjects.

I think it's largely bias. I tend to think more about elementary and middle school, and more about *mathematics* than about science. At the elementary school level, modelling and simulation are easier to integrate with science than with mathematics; *programming*, along with abstraction and algorithm, is easier to integrate with mathematics than with science.

Modelling, for example, is something that mathematics (and mathematicians) can *do*, and since mathematics can build models of mathematical ideas, modelling is also something that mathematics *uses*. But, at least as far as I see at the elementary school level (especially in the early grades) modelling *with* mathematics—creating mathematical models of phenomena—is very limited. And 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 reality, no kid 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's just what language *does*. By contrast, modelling 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 modelling.

Programming is exactly the opposite, easier to integrate into (early) mathematics than into science. (Of course, take this with a cup of salt, as I've 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 its input or adding 10 to its input, is easy programming. In fact, it's 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 *enact* the action and give feedback, which paper-pencil scrawl 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).

[5] Just as I was completing this paper, I received a copy of *Bebras* (Dagiené, et al., nd), a set of activities, many puzzle-like, that I found quite appealing, all designed to develop various elements of computational thinking in students.

[6] 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.

I'd *love* to get reaction to this last, very spur-of-the-moment rumination. What genuine *programming* activities, *at the elementary school level*, can be really well integrated with *science*? And what *modelling* or *simulation* activities, again at the elementary school level, can be really well integrated with mathematics?

Acknowledgments

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

References

BJC. (2017) The Beauty and Joy of Computing. http://bjc.edc.org

Cuoco, A. (1990) *Investigations in Algebra*. Cambridge, MA: MIT Press.

Dagiené, V., Stupuriné, G., Vinikiené, L., and V. Kincius. *Bebras*. Creative Commons Attribution-ShareAlike 3.0 Unported License (CCC BY-SA 3.0). http://www.bebras.lt

DESE. (2016) Massachusetts Department of Elementary and Secondary Education. *2016 Massachusetts Digital Literacy and Computer Science (DLCS) Curriculum Framework*. Accessed at http://www.doe.mass.edu/frameworks/dlcs.pdf, 16 September 2016.

EDC STEM+CT. (2018) http://go.edc.org/elementary-ct The Broadening Participation of Elementary Students and Teachers in Computer Science project is directed by Joyce Malyn-Smith. Accessed March 29, 2018.

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

Goldenberg. E.P. and Shteingold, N. (2007a) Early Algebra: The *MW* Perspective. In *Algebra in the Early Grades*, Kaput, J.J., Carraher, D.W. and Blanton, M.L., Eds. Erlbaum: Hillsdale, NJ, USA.

Goldenberg. E.P. and Shteingold, N. (2007b) The case of *Think Math!*. In *Perspectives on the Design and Development of School Mathematics Curricula*, Hirsch, C., Ed. NCTM: Reston, VA, USA.

Lewis, P. (1990) *Approaching Precalculus Mathematics Discretely*. Cambridge, MA: MIT Press.

Manousaridis, T. (2018), personal communication.

Mark, J., Goldenberg, E.P., Kang, J., Fries, M. and Cordner, T. (2014) *Transition to Algebra.* Heinemann: Portsmouth, NH, USA.

Noss, R. and Hoyles, C. (2018) The ScratchMaths project is directed by Richard Noss and Celia Hoyles, with Ivan Kalaš, Laura Benton, Alison Clark Wilson, & Piers Saunders. See http://www.ucl.ac.uk/ioe/research/projects/scratchmaths. Accessed March 29, 2018.

Otten, M., van den Heuvel-Panhuizen, M, Veldhuis, M., Heinze, A. and Goldenberg, P. (2017) Eliciting algebraic reasoning with hanging mobiles. *Australian Primary Mathematics Classroom*, Vol. 22, No. 3, pp. 14–19.

Papert, S. (1972) Teaching Children to be Mathematicians Versus Teaching About Mathematics, *Int. J. Math Ed in Science and Tech.*, Vol. 3, No. 3, pp. 249–262.

Sawyer, W.W. (2003) *Vision in Elementary Mathematics.* Dover: New York, NY, USA.

Schulz, L.E. and Bonawitz, E. B. (2007) Serious Fun: Preschoolers engage in more exploratory play when evidence is confounded. *Developmental Psychology*, Vol. 43, No. 4, pp. 1045–1050.

Wirtz, R., Botel, M., Beberman, M. and Sawyer, W.W. (1964) *Math Workshop*. Encyclopaedia Britannica Press: Chicago, IL, USA.

ISBN 978-609-95760-1-5