Susan Recommends

Gibbons, A. (2015) Debating digital childhoods: Questions concerning technologies, economies and determinisms. *Open Review of Educational Research*, 2(1), 118–127.

Feenberg, A. (1991). *Critical theory of technology* (Vol. 5). New York: Oxford University Press.

Plowman, L. (2016). Rethinking context: Digital technologies and children's everyday lives. *Children's Geographies*, *14*(2), 190–202.

Warschauer, M., & Matuchniak, T. (2010). New technology and digital worlds: Analyzing evidence of equity in access, use, and outcomes. *Review of Research in Education*, *34*(1), 179–225.

The New London Group. (1996). A pedagogy of multiliteracies: Designing social futures. *Harvard Educational Review*, *66*(1), 60–93.

Chapter 10

Coding as Another Language

Why Computer Science in Early Childhood Should Not Be STEM

Marina Umaschi Bers

What does computer science have to offer us in the 21st century? From smart watches to cell phones to automated cars, most of our objects have been programmed. They demand some basic understanding of cause and effect, the ability to sequence actions to achieve our goals, and the disposition to problem solve when things do not work as expected. Furthermore, algorithms dictate the news displayed in our social media, the people we might enjoy meeting and the merchandise we might want to purchase. If we do not understand what an algorithm is, we might not understand why and how certain information is or is not presented to us.

While we, as consumers of new technologies, need a basic skillset to exert some control over the smart artifacts in our lives, those of us who might become producers of these technologies, need more sophisticated engineering and computer science knowledge. However, all of us, consumers and producers, need to learn how to think in new ways about issues we have never encountered before. That is the true power of learning computer science: developing new ways of thinking about ourselves and our world.

Although most of us can identify the act of thinking and recognize its value, there is no scholarly consensus on its definition. What is thinking? It is the ability to make sense, interpret, represent, model, predict and invent our experiences in the world. It is facilitated by language. As Vygotsky wrote: "Thought development is determined by language, i.e., by the linguistic tools of thought" (Vygotsky, 2012, p. 100). Thus, as educators, we must give children one of the most powerful tools for thinking: language.

This is clearly understood in early childhood education. There is a strong focus on both building language skills and on helping children transition from oral language to written language. The teaching of literacy has occupied the field for a very long time (National Early Literacy Panel, 2008). Today, we have the opportunity to not only teach children how to think by using natural languages, but also by learning artificial languages – programming languages. Those are the languages understood by the smart objects and the algorithms that surround us.

I am using the term "language" to refer to a system of communication, natural or artificial, composed of a formal system of signs, governed by syntactic and grammatical combinatory rules, that serves to communicate meaning by encoding and decoding information. These systems of signs can be, for example, spoken, textual, graphical, gestural or tangible. This broad definition encompasses natural languages such as English, Spanish or Japanese, computer languages such as C or ScratchJr, sign language and tangible languages such as KIBO robotics. All of these have in common a limited set of signs that represent meaning and that can be combined in multiple ways following a specific set of rules to convey meaning.

The achievement of literacy with a natural language involves a progression of skills beginning with the ability to understand spoken words, followed by the capacity to code and decode written words, and culminating in the deep understanding, interpretation and production of text. The ultimate goal of literacy is not only for children to master the syntax and grammar, the orthography and morphology, but also the semantics and pragmatics, the meanings and uses of words, sentences and genres. A literate person knows that reading and writing are tools for meaning making and, ultimately, tools of power because they support new ways of thinking. In this paper, I am proposing that teaching coding ought to resemble the educational process used for teaching literacy. However, in current times, when most efforts to introduce coding take advantage of the growing push for STEM (Science, Technology, Engineering and Math) education, the powerful linkage with literacy and language gets lost.

"A literate person knows that reading and writing are tools for meaning making and, ultimately, tools of power because they support new ways of thinking."

The Problem with STEM

In the United States, the STEM acronym began to be used in education to address the perceived lack of qualified candidates for high-tech jobs. STEM came into the American consciousness in the 1950s to train the workforce and maintain national security. In 1958, during the height of the Space Race, the United States passed the National Defense Education Act, which provided funding and incentives for schools to improve both STEM and modern foreign language curricula (Kuenzi, 2008). Throughout different historical periods, the acronym went through variations in the order of its letters, such as SMET and MSTE, but finally settled as STEM.

As the cold war ended, the emphasis on national security diminished and the urgency to teach a foreign language dropped, while the need for a technically savvy workforce grew. In 2015, the STEM Education Act was passed,

the first time that federal funding for STEM was extended to cover computer science programs (Guzdial & Morrison, 2016). At the same time, well-funded nonprofits such as Code.org championed awareness and access to computer science in schools, by launching curricular initiatives, K-12 educational frameworks, professional development and policy changes.

The consolidation of STEM as a disciplinary cluster to fulfill the needs of the growing automated economy eclipsed the potential educational benefits of linking natural and artificial languages. As a result, schools isolated computer programming disciplinarily. It became the exclusive asset of STEM. However, things could have turned out differently. For example, what if instead of linking computer programming to economic growth and workforce preparation, it had been linked to intellectual growth and literacy education? What if the early argument had been that coding is about language? What if the pedagogies for teaching coding had borrowed teaching methods from literacy instead of math? What if the goal of teaching coding had been self-expression and communication, as opposed to problem-solving? Would this have allowed for a larger adult population that uses coding as a tool for creative expression and innovation and not just formulaic code writing? Would this have prevented the current lack of women and underrepresented minorities in the field? Would the cluster of STEM disciplines still own computer science? What if the role of computer science was conceived, from the beginning, not as a tool to educate the future workforce, but as a tool for thinking for the future citizenry? Throughout almost two decades, I have been pondering on these issues and developing technologies and pedagogical approaches to address the potential of teaching "Coding as Another Language" (CAL).

Coding as Another Language

Back in 1987, my mentor Seymour Papert called for the development of a field of study, which he called "computer criticism," by analogy with literary criticism. He wrote, "The name does not imply that such writing would condemn computers any more than literary criticism condemns literature ... The purpose of computer criticism is not to condemn but to understand, to explicate, to place in perspective" (Papert, 1987, p. 2). Papert envisioned that this new discipline would illuminate the role of computer programming in society and, most specifically, in education. Unfortunately, "computer criticism" never developed into a scholarly discipline, and the potential debate ended quickly by linking computer programming to STEM disciplines, instead of language arts or foreign languages. However, Papert and colleagues suggested that the process of learning to program may be akin to learning a foreign language in that they involve similar cognitive processes (Papert, 1980).

Although empirical research did not validate this observation, extensive work was done to design computational tools that reinforce the learning of

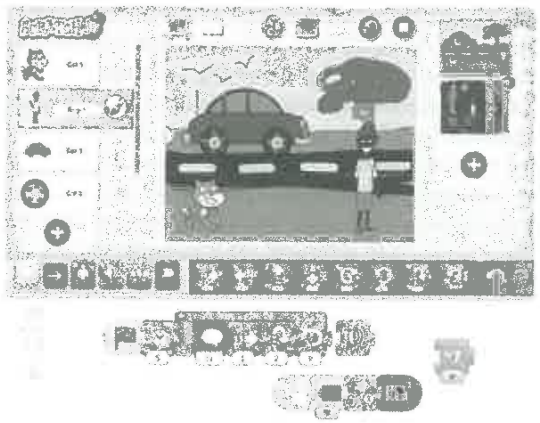

Photo 10.1 ScratchJr interface
Photo credit to Marina Umaschi Bers

Photo 10.2 KIBO robotic kit
Photo credit to Marina Umaschi Bers

coding as a literacy. For example, the DevTech Research Group that I direct at Tufts University has created programming environments for early childhood education, such as the free ScratchJr app (in collaboration with Mitch

Resnick at the MIT Media Lab) (see Photo 10.1) and the KIBO robotic system that uses tangible blocks instead of screens (see Photo 10.2), which teach coding with a literacy approach (Bers, 2018). In addition, our curricula, teaching materials and professional development strategies explicitly highlight the connection between the activity of coding and the mastering of a language and its uses to convey meaning. For example, in our *Dances around the World* curriculum, children learn to program their KIBO robots to dance to the beat of a culturally based traditional song and, at the end of the unit, invite family and friends to a robotic performance. While in the process of programming they learn powerful ideas from computer science and engineering, they engage in all steps of the design process, they put together their algorithms with the wooden KIBO command blocks and they debug their programs when things do not work, one of the most important activities they are doing is learning how to use a language to represent an idea and communicate something that is personally meaningful to them. I coined the term "Coding as Another Language" (CAL) to refer to our pedagogical approach.

CAL proposes that programming, as a literacy, engages new ways of thinking and new ways of communicating and expressing ideas – not only new ways of problem-solving. At a time when the US, among other nations, is struggling to understand if, how and when the teaching of computer science becomes mandatory, it is important to grapple with these pedagogical questions before policies are put in place. CAL understands programming languages as tools for thinking, and therefore, within this approach, learning to program as akin to learning how to use language.

A strong body of research in literacy education has helped us understand how young children learn to read and write and what instructional practices are most successful. Like any field of study there are dissenting camps, mostly rooted in differing beliefs about how literacy develops. Most famously, in the early 1990s, this debate led to the Reading Wars between proponents of a phonics emphasis in reading (cognitive theory) and proponents of whole language (maturation theory). But with increasing consensus, literacy scholars have embraced a "balanced diet" approach that agrees on the need for explicit phonic, phoneme and alphabetic strategy instruction, as well as authentic text and text-rich environments where discussion is central and student interpretations matter. Current understandings of literacy development provide a road map for how to transition students from oral language to written language through a series of fairly predictable stages (Chall, 1983; Duke & Pearson, 2002; Goldenberg, 1992). The assumption is that literacy is not a natural process like speech, where it unfolds in a child given the right conditions, but that it needs appropriate instruction, curriculum and assessment (Goldenberg, 2013). The same is true with coding.

Although the learning progressions for coding have not yet been as clearly identified as in literacy, work at my DevTech Research Group has contributed to describing the powerful ideas and skills in the continuum from

proto-programmers to expert programmers in early childhood (Bers, 2018). As with literacy, the coding progression does not just happen naturally; it requires instructional strategies. CAL understands the process of coding as a semiotic act, a meaning making activity, and not only a problem-solving challenge, even during its earliest, most basic levels of instruction. This understanding shapes how we develop our curriculum and our strategies for teaching coding.

The field of computer science education in early childhood has not yet had its Coding Wars, and yet there are two sides competing for attention. On the one side, there are those who provide introductory coding experiences through challenges to be solved with limited tools and languages. For example, Code.org popularized structured puzzle-like challenges for problem-solving tasks. Most of their lessons in the K-2 sequence feature a series of increasingly complex mazes that vary in theme but essentially rely on direction cues to move an object around the screen. From a cognitive perspective, there is sequencing and problem-solving, but this approach deprives children of the most powerful impact of literacy: expression of their own voices through the making of meaningful artifacts. This is the hallmark of the other side: those of us who align our work with Papert's Constructionism (1980) and who believe in immersing young children in computer programming languages, such as ScratchJr and KIBO, which are developmentally appropriate and, at the same time, provide opportunities for creating personally meaningful computational projects. Through this process, children learn to both code and use the code as a language for expression (Bers, 2018).

"From a cognitive perspective, there is sequencing and problem-solving, but this approach deprives children of the most powerful impact of literacy: expression of their own voices through the making of meaningful artifacts."

This process is akin to moving from "learning to read" to "reading to learn" (Carnine & Carnine, 2004; Chall, 1983). CAL integrates the teaching of coding with all of the early childhood curriculum, not only STEM. Children who learn to code can apply those thinking skills, abstraction, logic and problem-solving to everything else. Researchers in computer science education have coined the term "computational thinking" to refer to universally applicable attitudes and skillsets, rooted in computer science, that are fundamental for everyone to master (Wing, 2006, 2011). Coding engages and reinforces computational thinking. At the same time, computational thinking engages and reinforces coding. However, CAL proposes that when coding is introduced, thinking must also promote personal expression, communication and interpretation, and not only problem-solving.

"As more people learn to code and computer programming leaves the exclusive domain of computer science to become integral to other professions, it is more important than ever that we develop computer science pedagogies that promote deep and thorough engagement for everyone."

"As more people learn to code and computer programming leaves the exclusive domain of computer science to become integral to other professions, it is more important than ever that we develop computer science pedagogies that promote deep and thorough engagement for everyone."

Understanding computer science as a component of STEM education has restricted the power of coding to a limited group of disciplines, to a limited group of students and teachers and to the particular demands of the workforce. It diminishes coding's power as a true literacy that promotes new ways of thinking. If education aims at helping people think creatively to solve the problems of our world, only a subset of those problems can be solved by STEM disciplines. As more people learn to code and computer programming leaves the exclusive domain of computer science to become integral to other professions, it is more important than ever that we develop computer science pedagogies that promote deep and thorough engagement for everyone.

The field of literacy research has demanded that literacy instruction go beyond the important but insufficient benchmark of mechanical decoding and comprehension and should teach our children how to *use* their reading and writing as a tool for expression. CAL now moves the goalpost for computer science education. Our call is simple: let's teach coding to our children in a way that exposes it for what it is, a new language. Through this approach, the civic dimension of literacy comes into play. We are leaving the scribal age, when literacy was limited to a few chosen ones, and entering the printing press era, when it is available for the masses. As a literacy of the 21st century, coding has the power to bring about social change.

Marina's Essentials

The CAL approach embodies the following principles:

1. **Strategies used in literacy education can be helpful for teaching children how to code.**
2. **Coding projects can provide opportunities for children's sense making and expression of their own voices.**
3. **Problem-solving can serve as a means towards self-expression.**
4. **Coding activities can engage children in thinking about powerful ideas from computer science, as well as other domains.**
5. **The understanding of coding as an activity that engages children in using artificial languages to think in new ways.**

References

Bers, M. U. (2018). *Coding as a playground: Programming and computational thinking in the early childhood classroom.* New York, NY: Routledge Press.

Carnine, L. & Carnine, D. (2004). The interaction of reading skills and science content knowledge when teaching struggling secondary students. *Reading & Writing Quarterly, 20*(2), 203–218.
Chall, J. (1983). *Stages of reading development.* (2nd ed.). New York, NY: McGraw-Hill.
Duke, N. & Pearson, P. D. (2002). Effective practices for developing reading comprehension. In A. Farstrup & S. Samuels (Eds.), *What research has to say about reading instruction.* (3rd ed.). Newark, DE: International Reading Association, 205–242.
Goldenberg, C. (1992). Instructional conversations: Promoting comprehension through discussion. *The Reading Teacher, 46*(4), 316–326.
Goldenberg, C. (2013). Unlocking the research on English learners: What we know - and don't yet know - about effective instruction. *American Educator, 37*(2), 4–11.
Guzdial, M. & Morrison, B. (2016). Seeking to making computing education as available as mathematics or science education. *Communications of the ACM, 59*(11), 31–33.
Kuenzi, J. J. (2008). Science, Technology, Engineering, and Mathematics (STEM) education: Background, federal policy, and legislative action. *Congressional Research Service Reports,* **35.**
National Early Literacy Panel. (2008). *Developing early literacy: Report of the National Early Literacy Panel.* Washington, DC: National Institute for Early Literacy.
Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas.* New York, NY: Basic Books.
Papert, S. (1987). Information technology and education: Computer criticism vs. technocentric thinking. *Information Technology and Education, 16*(1), 22–30.
Vygotsky, L. (2012). *Thought and language.* Cambridge, MA: MIT Press.
Wing, J. M. (2006). Computational thinking. *CACM Viewpoint,* 33–35. Retrieved from www.cs.cmu.edu/afs/cs/usr/wing/www/publications/Wing06.pdf.
Wing, J. M. (2011). Computational thinking. Presented at *IEEE Symposium on Visual Languages and Human-Centric Computing,* 3.

Learn More about Marina's Work

- *Coding as a Playground: Programming and Computational Thinking in the Early Childhood Classroom,* Routledge (2018).
- *The Official ScratchJr Book: Help your Kids Learn to Code,* No Starch Press (2015).
- *Designing Digital Experiences for Positive Youth Development: From playpen to playground,* Oxford University Press (2012).
- *Blocks to Robots: Learning with Technology in the Early Childhood Classroom,* Teachers College Press (2008).

Marina Recommends

- The ScratchJr app http://scratchjr.org
- The KIBO robot http://scratchjr.org

Chapter 11

Personalized Education and Technology

How Can We Find an Optimal Balance?

Natalia Kucirkova

Introduction

Personalized versus standardized education: which one do you prefer? That's a loaded question for sure, since standardized education has become associated with inflexible accountability measures for teachers and testing overload for children. With the renewed interest in creativity and child-centered learning, standardized education seems to present the perfect foil for the adoption of personalized education: personalized education promises to offer tailored and flexible frameworks for teachers and choice-led curriculum for children.

With the advent of affordable and ubiquitous personal mobile devices, personalized education became the new kid on the educational reform block. Technology has always been part of the interest to transform old educational paradigms. However, the personalized versus standardized education dichotomy is misplaced: to foster holistic outcomes in complex environments, we need combined, not reduced, versions of education. In this essay, I will try to convince you that we need both sides of the educational coin and both strong design and pedagogy.

First, we need to agree that technology-driven education might sound seductive to technology investors and some disillusioned professionals, but it cannot be the driving force in sustainable education models. Technology-driven education emphasizes design rather than pedagogy. Such myopic approaches to technology deployment are characterized by large investments made into hardware rather than professional training and pedagogical support of technology use. My education alarm goes off every time I hear a company claiming to have developed a product which will "transform" children's learning overnight. In my work and in this chapter, I take the approach that technology can mediate and enrich, but not replace or determine, teachers' expertise in the classroom.

Technology-Mediated Education

The role of technology in supporting standardized education is clear: broadly speaking, technology can support organized progression of students across