Guest editorial

Learning to code, coding to learn: youth and computational thinking

Introduction

Professor Jeannette Wing's provocative and influential paper entitled *Computational Thinking* appeared in the March 2006 issue of Communications of the ACM; in the 13 years since, educators, computer scientists, policy makers, and technologists have been working to define this conceptual space, measure it, and assess the role that computer science can and should play in the education of young people. Although Wing is by no means the first person to notice that computer science can play an important role in developing problem solving capacities in youth across the curriculum (Papert, 1980; Clements and Gullo, 1984; Harel and Papert, 1990; diSessa, 2001, to name just a few), her call to arms fuelled increasing research attention and policy interest (Aho, 2012; Cooper and Cunningham, 2010; Guzdial, 2008; Wang, 2015, Wing, 2008; Wing and Stanzione, 2016). With this special issue of the *Journal of Information and Learning Sciences*, we propose that computational thinking (CT) is a generative space residing between the learning sciences and information sciences, drawing on concepts of cognition and development (e.g. motivation and self-regulation), the system sciences (e.g. algorithmic representation and design of data structures) and areas of shared or interdisciplinary concern and interest (e.g. digital literacy, problem-solving, making and creativity).

But, what is CT? And how does it relate to critical thinking, not to mention pressing issues of digital literacy, and children's readiness to engage with contemporary digital media? Succinctly, CT can be defined as "the process of recognising aspects of computation in the world that surrounds us, and applying tools and techniques from Computer Science to understand and reason about both natural and artificial systems and processes" (Royal Society, 2012, p. 29). Further definitions, including those offered by the College Board and National Science Foundation, identify key thought processes and techniques that help us operationalize CT as a set of practices that can be taught and fostered, including abstraction and pattern generalization; systematic processing of information; symbol systems and representation; algorithmic notions of flow and control; structured problem decomposition; iterative, recursive, and parallel thinking; conditional logic; efficiency and performance constraints; debugging and systematic error detection (Grover and Pea, 2013). This list shares a number of common points with the curriculum taught in mathematics and science; however, there clearly are procedures and habits of mind, such as abstraction and debugging, that are distinctive and not likely to be taught in general K–12 courses (Wang, 2015). CT has some overlap with general problem-solving processes that form the basis of information literacy models as well, but the emphasis on understanding algorithmic flow and symbolic representation suggests significant differences. Scholars are making the case that CT is its own form of literacy, one that complements other knowledge domains, but occupies its own place in the curriculum (Barr and Stephenson, 2011; Guzdial, 2008).

In this editorial article, I will briefly, though not exhaustively, review research from diverse perspectives on CT, first from education and the learning sciences, then from library and information science. I will then opine on the challenges confronting society by applied computer science that lacks a consideration of diverse human values and propose a way forward that relies an ethically grounded education in CT. I conclude with brief summaries of the articles that represent this special issue and additional areas of future consideration that the editors look forward to seeing published in this journal and other venues.

Information and Learning Sciences
Vol. 120 No. 5/6, 2019
pp. 254-265

2398-5348
DOI 10.1108/ILS-05-2019-139

Perspectives from education and the learning sciences

In recent years, the Computer Science Education (CSE) movement has gained considerable momentum, led by a coalition of scholars, non-profits and industry partners. Coding interfaces such as MIT's Scratch platform, Gamestar Mechanic, Kodu, and a host of others (Anton and Berland, 2014; Resnick *et al.*, 2009) have opened new possibilities for youth to develop their own interactive games. The "Computer Science for All" Initiative (CS4All) begun during the Obama Administration suggests that the USA is not far behind France, the UK and other nations in mandating coding for children beginning in the elementary grades. Programs and initiatives in the North American context that contribute to these efforts include Code.org, Hour of Code, and the work of organizations including BlackGirlsCode, GirlsWhoCode, iRemix, Code Savvy, Globaloria, KidsCodeJeunesse and others.

Scholars in formal and informal learning have been working to make computer programming more accessible to young people. According to a recent survey, coding is already a part of the formal curriculum of 16 countries in Europe (Balanskat and Engelhardt, 2015). Curricula in game design, such as those developed by constructionist scholars and instructional design experts Yasmin Kafai, Idit Harel and their colleagues (Kafai, Peppler and Chapman, 2009; Fields *et al.*, 2012; Reynolds and Harel, 2011; Reynolds, 2016) have engaged thousands of young people across several US states in formal, intensive in-school introductory CSE coursework.

Although these programs show promise, their over-emphasis on "games" as the endpoint of computer programming may be perceived as limiting. The Raspberry Pi, for example, was developed in part as a response to computer science majors at a university having computer skills limited to gaming and web design and not understanding simple engineering concepts that make hardware and software work together. Coding is increasingly linked with making, a movement that has engaged curious learners of all ages and backgrounds. Making as a way of fostering CT often expands into physical computing, where software and hardware come together, and learners "tinker" with wires, sensors, LEDs, and other components. Learning through physical computing has the potential to generate a more holistic approach to CT, involving reasoning about a wider range of systems and processes. New curricula are emerging inspired by the confluence of coding and making (Honey and Kanter, 2013; Sheridan *et al.*, 2014) and the "connected learning" work of Mimi Ito *et al.* (2013).

Perspectives from library and information science

Although CSE might be perceived as in a renaissance of sorts, the literature in library and information science has given only modest attention to CT as an outcome of library-based informal learning programs. This is in part due to the historic focus on young people as users of information systems, rather than creators of information systems. The emphasis on system use, and in particular search systems and social media services, often casts young persons in one of two lights. In the first, we have the glamorization of the youth technology user, the digital native, fettered by the constraints of school structures and adult perceptions, a "power user" by nature whose talents we are only beginning to understand (Prensky, 2001; Palfrey and Gasser, 2008). In the second, what some call the deficit model, young people are described in unflattering terms as digital "naïves", ignorant and cavalier users, the "Google Generation", whose focus on expedience and ease of use often leads to poor information processing and diminished learning outcomes (Carr, 2011). The reality is somewhere between these two characterizations (Boyd, 2014; Meyers *et al.*, 2013; Wagner, 2008).

One way that CT is gaining ground in LIS scholarship is through research on the role of makerspaces as a *locus* of creative and critical skill-building for youth. As libraries have adopted "making" as an informal learning opportunity to engage preteen and teens, the links to coding and computational skill-building have followed in work by several LIS

researchers (Abbas and Koh, 2015; Bowler and Champagne, 2016; Koh *et al.*, 2019; Martin, 2015; Prato, 2017; Subramaniam *et al.*, 2018). Informal science, technology, engineering and mathematics (STEM) education through public libraries can complement school-based computational initiatives, providing interest-driven engagement for youth of all ages. Of note, these scholars, particularly Bowler and colleagues, maintain a twin focus on youth empowerment and critical engagements with technology. This area of scholarship provides a fruitful area of cross-domain development for scholars in LIS and the learning sciences.

A challenge for learners of all ages is the "black boxing" of technology, which makes contemporary tools and media easy to use and yet difficult to comprehend. Few undergraduate students, for example, even in Masters of Information Science courses, understand the difference between Google search and the contemporary online library catalogue, even as the latter is trying to look and function like the former. CT, then, supports critical understanding at the intersection between system design and system use, providing the vital link to comprehending *how* information systems work.

Coding and design: Connecting the analogue and digital

To connect CT with students' everyday lives, it can be useful to create functional analogies that help demystify computing. Computer science does not have to be taught exclusively with digital tools, as evidenced by the CS unplugged curriculum (https://csunplugged.org/) and related research (Taub *et al.*, 2009). In my own teaching in information science, I often encounter students who are apprehensive about coding and lack awareness of how information systems work. To further elaborate on how we can break down the "black boxes" of technology, I offer the following example.

Elementary school students can be found constructing and using "cootie catchers" (also known as fortune-tellers or chatterboxes), folded paper manipulatives that function much the same as a "magic 8-ball" device. Students invariably recognize this object but may not connect it to information concepts immediately. The idea behind these paper devices is to craft a series of scenarios or "fortunes" that might describe any number of persons or contexts, much like the slip inside a fortune cookie. The paper hand manipulative serves as a means by which the fortunes are delivered to persons on the auspices that they "select" a fortune by indicating a personal detail, a favourite colour or number. What consistently comes as a surprise to learners is that this device is a simple prototype of an information system. The hand manipulative is the interface; the favourite number or colour selected by the user becomes input for the algorithm; the "fortunes" written in advance inside the flap of paper constitutes a database. Items in the database are called via the interface through individual interaction with the algorithm. In a matter of minutes, we can assign these sophisticated concepts to familiar aspects of a child's social world, crafted from paper and markers.

The power of crafting can then be translated to a digital iteration of the "cootie catcher" system through the programming of a fortune-teller in Scratch, an object-based programming language readily available on the web or installed as an app. The power of CT is unleashed when the learner makes the connection between the manipulative and the code, the physical artifact and the digital system. This is what I seek to do with this lesson: learners connect crafts and activities with the concepts of systems design and development. In the process, learners are prompted to identify the shortcomings of the physical system (i.e. your database is limited to how many flaps your cootie catcher has) and how the digital system might overcome these challenges or create new ones. As my students explore the connections between physical and digital information systems, they come to understand how our contemporary systems work and the role that individual choices play in how systems function, both as designers and users. The "cootie catcher" database is just one example of how we can connect and unpack user behaviours and

information systems to promote CT. These activities foster reflection in combination with coding skills, so students can see what they are doing and why it makes sense. I argue that only in the process of building their own information systems can users make the black box of today's information infrastructure transparent. In doing so, they find the middle ground between users and designers, coders and end-users.

(Not) Thinking like (certain) computer scientists

I teach children and adult students to code, and I believe that CT can be a powerful tool in creating future systems that support vibrant communities and rich human interactions. Yet, recent revelations regarding the use of computing platforms and technologies give me and others cause for alarm. It is worth taking a moment to interrogate the challenges we see in contemporary technologies and identify the place of CT in moving toward a future where technology supports, rather than hinders, human thriving.

"To a computer scientist, the solution to a bug is often just more computer science", noted Brooke Borel (2018), writing about the challenges of "Deep Fake" video creation. Deep fake technology, or image synthesis using artificial intelligence, is emerging as a development both useful and pernicious. The product of deep learning algorithmic mash-ups of audio and video, the technology will soon produce near perfect video of anyone saying anything, provided there is sufficient high-quality material to train the system. Although this technology was intended to streamline cinema production and enhance long-distance communication, it could also be used by individuals, groups or governments to create propaganda, misleading or slanderous political ads, or personally damaging videos that could be uploaded to social media (Chesney and Citron, 2018). The computer scientists who invented these techniques, however, suggest figuring out how to regulate this technology is someone else's problem. Or, they suggest, maybe we can invent algorithms to identify when someone is misusing the algorithms (Adler, 2018).

Our news cycle is now filled with dramatic and painful examples that illustrate how this particular computer science mindset – that the solution to emergent problems of code is the development of more and increasingly powerful counter-code – is fundamentally flawed. Algorithms contribute to the suppression of dissenting minorities, live-streaming of terror attacks, racial profiling and discrimination and efforts toward ethnic genocide (Eubanks, 2018; Noble, 2018; Buckley *et al.*, 2019). Little did we realize from the cheeky early moto of Facebook – move fast and break things – that the "things" being broken or subverted might include privacy laws, cultural norms and human decency (Toyama, 2015; Taplin, 2017). Although it is also true that algorithms are solving problems, saving lives and creating cleaner, safer living conditions for citizens around the world, technology companies need to take ownership of the failures as well as the successes. We need to ensure that there is a strong ethical component to coding curricula that facilitates thinking through the consequences, intended or unintended, of computer science. Code is a proximate cause of the unfortunate consequences of its (mis)use, but like guns and cars, everyone can likely agree that powerful tools needs oversight and regulation, as well as training that establishes ethical applications thereof.

This is not to suggest that computer science as a discipline is ethically bankrupt; far from it. Many of us benefit daily from the technical advances of computation, and algorithms promise new ways of realizing human potential when applied to any number of pressing geopolitical issues, including climate change, sustainable agriculture, women's rights, and health care to name a few. Nor am I conflating the discipline of computer science with technology industries, although evidence that portions of the academy are increasingly under the sway of corporations is a significant concern (Giroux, 2007). Yet, we have come to realize that the largest players in the technology industries, such as Facebook, Google and Amazon, are incapable of regulating themselves; they appear to have chosen to focus on profits at the

expense of human thriving (Hughes, 2019; Susser, 2019). These companies, among the largest, most valuable corporate entities on Earth, are also key players in the development and propagation of free coding curricula for schools and individuals. If you have experienced a coding massive online open course (MOOC) or series of lessons on Code.org, you will find the assumptions and ethos of Silicon Valley are baked in. To create a check against tendencies to code first and ask questions later, we need to ensure that human values are at the centre of our thinking about code, its use and its consequences.

Toward critical computational literacies

Code, like any human language, is not value neutral. Both the building blocks of computation as well as what we construct with those blocks carry human values. Those values may be efficiency, parsimony, safety or accountability. How we define those values and make sense of them is an essential part of coding education. Learning to use the language of code is also a value proposition, as code facilitates entering a conversation as well as a community of other coders. The ethics and values of that community play a part in how we perceive technology and each other in the process of its use and deployment in the world.

Several of the authors in this special issue push back on framings of CT, and I am encouraged by their development and use of different terms to describe how people think with computers and code for diverse outcomes. In particular, Wells and colleagues propose the term *procedural creativity* to describe the process of game-based engagements with youth. Recently, Lee and Soep (2016) have combined an emancipatory philosophy with CT to coin the phrase "critical computational literacy." My own work has taken a turn toward critically engaging with questions of how we use technology and how coding is represented, both in curriculum that supports CT, as well as the popular imagination. Let me provide two examples.

Through a series of workshops administered in local public libraries, my students and I have been introducing preteens and teens to computer science concepts using the Raspberry Pi, taking advantage of its built-on support for introductory coding (Scratch, Python, and SonicPi) and physical computing (Meyers, 2016). We took these activities on the road to work with urban teens in an all-day workshop sponsored by the Free Library of Philadelphia. With colleagues from several universities, we combined a physical computing activity, a design workshop, social interaction and play to focus on social justice, and how technology could help them improve their communities (Fisher *et al.*, 2017). Before jumping into the activity, we reflected on the digital cameras they use every day and how little they understood about how they work. Next, teens used readily available electronics (such as camera modules, jumper wires and bread boards) and a Python script to build a working camera using the Raspberry Pi as a base. Youth worked in pairs; adult facilitators roved and provided guidance. A set of printed instructions broke the project into steps and prompted the teens to check their progress along the way. The goal was to help students reflect on the process of making technology and empower/inspire them as designers, coders and makers. After the build, teens engaged in a design workshop where they envisioned prototype technologies that could contribute to a better community. Their designs were simply astounding. What struck the researchers was how many of the designs focused on building empathy and social understanding – seeing the world through their eyes – and making systems that fostered peace, justice and accountability.

Most recently, I have been analyzing children's stories and media that incorporate computational elements, either implicitly or explicitly, in the plot of the narrative. For example, I interrogate Gene Luen Yang's *Secret Coders*, a series of six graphic novels (published 2015–2018) that use learning to code as an essential element of its narrative structure. The novels follow Hopper, Eni and Josh, three preteen outsiders who initially use their growing knowledge of computer coding to learn about their mysterious school, Stately Academy. Eventually, these

skills are used to save the world from the nefarious Professor One-Zero and his formula for "true happiness." I argue that Yang's series contains elements that play to Silicon Valley's Randian, pro-technology motives, while also subverting them, enabling readers to critically engage with the discourses of technology and its role in society (Meyers, 2019 forthcoming). This analysis is part of a larger examination of the rhetorics of computation present in children's everyday media and draws in other texts such as popular cinema (e.g. Big Hero 6), picture books (e.g. Hello Ruby), and children's games (e.g. GoldieBlox). As part of this investigation I ask the following: whose story is actually being told? How are children silenced or empowered through narratives extolling the power of computation? When discussing children's texts, we often look for the "hidden adult" (Nodelman, 2008), by which we seek to unpack the author's relationship to the subject and reader, and interpret stories as social texts. One might argue that the hidden adult in Yang's graphic novels is an entire industry: Silicon Valley and its neo-liberal agenda, which is using coding education and rhetorics of computation as a kind of "political technology" to shape both minds and social systems (Meyers, 2019 forthcoming). In addition to critically reading and analyzing these texts, my students and I are developing techniques for co-reading/co-viewing/co-playing with these media. We hope through this work to bring a needed critical eye to computing education, as well as empower young people to see the element of computation in the world around them and both celebrate *and* interrogate their value.

The articles in this issue

CT, as a way of framing the what, how and why of coding and related 21st-century skills, is a popular and generative topic. Our call prompted a range of submissions covering work with younger children through post-secondary CSE, using both qualitative and quantitative methods, as well as conceptual approaches. In the following text, I give a brief introduction to each article in this issue and conclude with areas for future study, including those topics we wish we could have included here.

Matthew Wells and Jason Boyd push back on the term "computational thinking" in their article *Generating Gameworlds: The Case for Procedural Creativity*. They pose this question to readers: is CT the best way to describe the student outcomes we seek to achieve though coding and computational activities in and out of school? They note a key shortcoming of CT, especially as professed by computer science educators, is the emphasis on abstraction and solutionism. Their proposed framework, procedural creativity, is more closely aligned with the language-action perspective, and we can see in their elaboration how it shifts computational activity from the cognitive (rational) position to a situated, socio-cultural standpoint. They propose that game generation, as well as the use of interactive fiction, serves as an ideal platform for engaging this concept. Their work dovetails nicely with the work of Proctor and Blikstein described in the following text.

In agreement with Wells and Boyd, Chis Proctor and Paulo Blikstein illustrate how interactive narrative may serve both a pedagogical and emancipatory function in *Unfold Studio: Supporting Critical Literacies of Text and Code*. Readers are introduced to Unfold. studio, an interactive storytelling platform designed to foster "textual-computational multiliteracy." Through several workshops, first to design the platform and then to test it with a diverse group of high school students, Proctor and Blikstein used a designed-based research approach that takes us from localized innovation toward a replicable pedagogy of critical literacy. The authors use the case of one student, Leanne, and her composition "Angela: A Bystander's Story" to unpack the potential of interactive storytelling to elucidate the interaction between discourse (language) and discourses (ideologies). The work also demonstrates how critical computational literacy can be situated in both formal and informal learning environments.

Also working in the classroom, Eben Witherspoon and Christian Schunn examine the relationship between teachers' dispositions toward CT and student performance in robotics curricula. When teachers emphasize CT as a learning goal, students not only show greater knowledge gains but also maintain more positive attitudes toward programming over time. As teachers may come to computational curricula and skills through means other than traditional pre-service training in computer science (e.g. in-service training, workshops, or self-directed learning via open resources), it is vital that researchers pay attention to how teachers deliver lessons in coding and computational problem-solving, not just how student attitudes influence their reception. As I mention later in this paper, more attention is needed on teacher training, particularly for classroom educators and informal learning facilitators, such as librarians, who are already in the field.

Focusing on informal learning, Ricarose Roque and Natalie Rusk share qualitative insights on coders' practices in *Youth Perspectives on Their Development in a Coding Community*. They interviewed eight deeply involved members of the Scratch community to understand their motivations and engagements in coding. The interviews emphasized memorable moments – those pivotal interactions, conversations or projects – that participants identified as key to their trajectories in the community and their development as "coders." Although a handful of highly-engaged participants is not meant to represent a broader sample, these success stories give us an opportunity to analyze what works (and what does not) in a diverse online community of learners. The results show how feedback and recognition, as well as opportunities to mentor others, may drive achievement in online learning platforms, which is crucial to understanding the role of informal learning networks that support life-long, life-wide learning.

Paula Haduong unpacks the reluctance of teens to engage in coding education in *"I Like Computers, I hate coding": A Portrait of Two Teens Experiences*. Using the technique of portraiture, Haduong brings greater depth and meaning to the disconnect young people may feel between everyday computer use for communication and creative expression and the seemingly onerous task of writing code. Exploring concerns for motivation in the design of informal computational learning opportunities, particularly for women and underrepresented minorities, the authors identify the diverse expectations youth bring to such programs and how identity and social support affect learner experiences. The two portraits of youth developed in this piece give us a great deal to think about in terms of how we might make computer science education more equitable. The authors suggest that the solution is not strictly about providing more access to coding for persons of colour; rather, there are systemic inequities that need to be overcome before some persons of colour will want to engage in coding education.

Balancing these intimate, qualitative portraits of young coders, we are delighted to include a mixed-methods analysis of young women coders and their longitudinal trajectories in STEM fields. Seeking to identify paths to "persistence" as a way of realizing greater equity in computational and STEM-oriented occupations, Joanna Weidler-Lewis, Wendy DuBow, Alexis Kaminsky and Tim Weston provocatively build an argument for compulsory coding education in K–12. Their work also reinforces the notion, albeit with different evidence than Hanuong, that persistence in coding is multifaceted and is spurred by person ambition, family dynamics and social support structures. Although many can agree that greater diversity in the technology industries would be an essential good, we must also consider diverse outcomes for people who want to study computer science, such as applying coding skills in education, public policy and innovation in the non-profit sectors.

On the quantitative side of the conversation, Ömer Demir and Süleyman Sadi Seferoğlu introduce readers to their efforts to design a Scratch-based coding achievement test for

students in higher education in Turkey. As Scratch is widely used across education levels, and has set the standard for block-based coding education, we anticipate this paper will draw attention from researchers and practitioners alike. This work builds on prior efforts to construct validated tests, but with the key distinction that this new assessment is not coding language independent. The authors argue that prior work to establish computational literacy tests that are language neutral tend to measure algorithmic and logical processes, rather than contextualized application skills. We need robust ways of measuring instructional effectiveness for computational literacy, both for informal and formal learning, and this research project is a step in that direction.

Despite the richness of approaches to the theme of "learning to code, coding to learn" presented here, we acknowledge that there are several areas that could see further research development, and we encourage and look forward to work that addresses these subtopics:

- *Adult and non-traditional learners*: The emphasis of many coding programs, such as Girls Who Code and Kids Code Jeunesse, are youth, and more specifically pre-teens, teens who are yet to find employment. Yet, coding is also an essential competency that many adults find important once they have left school or are looking to develop for personal advancement, for re-skilling and employment opportunity, or simply to be more effective in an increasingly high-tech workplace. The locus of this training may be online tutorials via YouTube or Lynda.com, evening classes at a community centre or public library, coding MOOCs via EdX or tinkering with like-minded makers at weekend meet-ups. My colleague Huang Hong has been exploring "live coding" communities where people code in real time, and there is a great deal to explore in terms of how these communities function as performance spaces, as well as personal and professional development (Haaranen, 2017). Chilana *et al.*'s (2016) work on "conversational coding" is another interesting area of investigation. In this work, Chilana and colleagues unpack the motivations and strategies of people who want to learn coding to better engage with co-workers who do software development, even if they have no intention of compiling and running a program themselves in the future. Coding happens across the lifespan, and many demographic groups could benefit from computer science education at different levels of sophistication.
- *Teachers and librarians*: As CT is embedded in more curricula and supported through activities in diverse knowledge domains as language arts, science, and social studies, it will become essential to train more educators in the basics of coding. To deliver on the promise of CS4All, teachers need to be confident enough with technology to open pathways for learners, even if they are never expert coders themselves. This will require upgrading teacher education programs, as well as providing extensive professional development for in-service educators. The American Library Association's "Ready to Code" (RtC) initiative (www.ala.org/tools/readytocode/home) is one such program that seeks to build capacity for in-service school and public librarians by seeding local initiatives in libraries, as well as increasing the use of computational concepts in post-secondary library training. Programs like RtC are nascent but already showing signs of progress; yet, there is still a great deal to do in this area.
- *Early childhood*: An explosion of coding-related toys and games have entered the marketplace in recent years, some of which have tactile or manipulative elements, that target early learners from ages of 3 to 8 years. These toys may be stand alone or work in conjunction with screens, such as the iPad, to encourage early computational play. The Hour of Code (https://hourofcode.com/) initiative has encouraged schools to adopt a number of free or low-cost coding apps (e.g.,

LightBot, Tynker, Kodable, and Scratch Jr.) that target pre-school and early elementary audiences. Despite the amazing diversity of entries we are witnessing, there has been little research that validates the utility or efficacy of these toys for coding education. Parents and teachers are challenged in deciding which of these, if any, are worth investing in, or building curriculum around, and serious reviews and comparisons are difficult to find. One means by which libraries can support these decisions is through media mentorship (ALSC, 2015) and the development of maker spaces and technology "petting zoos," where you can try before you buy.

These three focus areas are certainly not the only areas where work in needed, but the editors of this special issue call these out for their tremendous opportunity for additional research and potential to affect new audiences for computational education and training.

Conclusion

Fishman *et al.* (2004) identified that innovations involving learning require capability, culture and policy to scale beyond initial efforts. We look forward to the future work of the authors in this issue as well as remind ourselves of the significant work that is yet to come. CT can be approached from a cognitive position as a type of computational problem-solving, with attendant skills, knowledge and dispositions that frame the work of coding. However, a socio-technical perspective acknowledges that computational skills and dispositions are deeply entwined with other concerns, including identity, social infrastructure, public policy, ethics, and human values. I argue that code is a political technology, one that we cannot and should not disentangle from the applications of computing, and the social contexts of use. If we seek to empower learners young and old through CT, procedural creativity, or critical computational literacies, our first step is to envision how these tools can contribute to human thriving. And if we find that our tools lead us away from a centring on human well-being, perhaps the greatest lesson is knowing when to seek a non-computational resolution, that is, knowing when *not* to code.

Eric M. Meyers
School of Information, University of British Columbia, Vancouver, Canada

References

Abbas, J. and Koh, K. (2015), "Future of library services supporting teen learning: Perceptions of professionals in learning labs and makerspaces", *Journal of Research on Libraries and Young Adults*, available at: www.yalsa.ala.org/jrlya/2015/11/future-of-library-and-museum-services-supporting-teen-learning-perceptions-of-professionals-in-learning-labs-and-makerspaces/ (accessed 14 February 2019).

Adler, S. (2018), "Breaking news" | Radiolab, available at: www.wnycstudios.org/story/breaking-news (accessed 2 January 2019).

Aho, A.V. (2012), "Computation and computational thinking", *Computer Journal*, Vol. 55 No. 7, pp. 832-835.

Anton, G. and Berland, M. (2014), "Studio K: a game development environment designed for gains in computational thinking", paper presented at the 45th ACM Technical Symposium on Computer Science Education, Atlanta, GA.

Association for Library Service to Children (ALSC), Campbell, C., Haines, C., Koester, A. and Stoltz, D. (2015), *Media Mentorship in Libraries Serving Youth*, American Library Association, Chicago, IL.

Balanskat, A. and Engelhardt, K. (2015), "Computing our future: computer programming and coding - priorities, school curricula and initiatives across Europe", available at: www.eun.org/documents/411753/817341/Computing+our+future_final_2015.pdf (accessed 12 August 2017).

Barr, V. and Stephenson, C. (2011), "Bringing computational thinking to K-12: what is involved and what is the role of the computer science education community?", *ACM Inroads*, Vol. 2 No. 1, pp. 48-54.

Borel, B. (2018), "Clicks, lies and videotape", *Scientific American*, Vol. 319 No. 4, available at: www.scientificamerican.com/article/clicks-lies-and-videotape/ (accessed 12 January 2019).

Bowler, L. and Champagne, R. (2016), "Mindful makers: question prompts to help guide young peoples' critical technical practices in maker spaces in libraries, museums, and community-based youth organizations", *Library and Information Science Research*, Vol. 38 No. 2, pp. 117-124.

Boyd, D. (2014), *It's Complicated: The Social Lives of Networked Teens*, Yale University Press, New Haven, CT.

Buckley, C., Mozur, P. and Ramzy, A. (2019), "How China turned a city into a prison: a surveillance state reaches new heights", *New York Times*, 4 April 2019, available at: www.nytimes.com/interactive/2019/04/04/world/asia/xinjiang-china-surveillance-prison.html (accessed 5 April 2019).

Carr, N. (2011), *The Shallows: What the Internet Is Doing to Our Brains*, W.W.Norton, New York, NY.

Chesney, B. and Citron, D. (2018), "Deep fakes: a looming challenge for privacy, democracy, and national security", *California Law Review*, available at: https://ssrn.com/abstract=3213954 (accessed 12 January 2019).

Chilana, P.K., Singh, R. and Guo, P.J. (2016), "Understanding conversational programmers: a perspective from the software industry", *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*, *ACM*, *New York, NY*, pp. 1462-1472, available at: https://doi.org/10.1145/2858036.2858323

Clements, D.H. and Gullo, D.F. (1984), "Effects of computer programming on young children's cognitions", *Journal of Educational Psychology*, Vol. 76 No. 6, pp. 1051-1058.

Cooper, S. and Cunningham, S. (2010), "Teaching computer science in context", *ACM Inroads*, Vol. 1 No. 1, pp. 5-8.

diSessa, A.A. (2001), *Changing Minds: Computers, Learning and Literacy*, The MIT Press, Cambridge, MA and New York, NY.

Eubanks, V. (2018), *Automating Inequality: How High-Tech Tools Profile, Police, and Punish the Poor*, St. Martin's Press, New York, NY.

Fields, D.A., Searle, K.A., Kafai, Y.B. and Min, H.S. (2012), "Debuggems to assess student learning in e-textiles", paper presented at the 43rd SIGCSE Technical Symposium on Computer Science Education, Atlanta, GA.

Fisher, K.E., Martens, M., Peterson-Kemp, A. and Meyers, E.M. (2017), "Ray of sunshine happiness gun' and other apps in pursuit of social justice: teens' designs from Philadelphia", paper presented in iConference 2017 – Effect • Expand • Evolve: Global Collaboration Across the Information Community, March 22-25, Wuhan.

Fishman, B., Marx, R.W., Blumenfeld, P., Krajcik, J. and Soloway, E. (2004), "Creating a framework for research on systemic technology innovations", *The Journal of the Learning Sciences*, Vol. 13 No. 1, pp. 43-76.

Giroux, H.A. (2007), *University in Chains: Confronting the Military-Industrial-Academic Complex*, Paradigm Publishers, New York, NY.

Grover, S. and Pea, R. (2013), "Computational thinking and K-12: a review of the state of the field", *Educational Researcher*, Vol. 42 No. 1, pp. 38-43.

Guzdial, M. (2008), "Education paving the way for computational thinking", *Communications of the Acm*, Vol. 51 No. 8, pp. 25-27.

Haaranen, L. (2017), "Programming as a performance: live-streaming and its implications for computer science education", *Proceedings of the 2017 ACM Conference on Innovation and Technology in*

Computer Science Education (ITiCSE '17), *ACM*, *New York, NY*, pp. 353-358, available at: https://doi.org/10.1145/3059009.3059035

Harel, I. and Papert, S. (1990), "Software design as a learning environment", *Interactive Learning Environments*, Vol. 1 No. 1, pp. 1-32.

Honey, M. and Kanter, D.E. (2013), *Design, Make, Play: Growing the Next Generation of STEM Innovators*, Routledge, New York, NY.

Hughes, C. (2019), "It's time to break-up Facebook", *New York Times*, 9 May 2019, available at: www.nytimes.com/2019/05/09/opinion/sunday/chris-hughes-facebook-zuckerberg.html (accessed 9 May 2019).

Ito, M., Gutierrez, K., Livingston, S., Penuel, B., Rhodes, J., Salen, K., Schor, J., Sefton-Green, J. and Watkins, S.C. (2013), *Connected Learning: An Agenda for Research and Design*, Digital Media and Learning Research Hub, Irvine, CA.

Kafai, Y.B., Peppler, K.A. and Chapman, R.N. (2009), *The Computer Clubhouse: Constructionism and Creativity in Youth Communities*, Teachers College Press, New York, NY.

Koh, K., Snead, J.T. and Lu, K. (2019), "The processes of maker learning and information behavior in a technology-rich high school class", *Journal of the Association for Information Science and Technology*, forthcoming.

Lee, C.H. and Soep, E. (2016), "None but ourselves can free our minds: critical computational literacy as a pedagogy of resistance", *Equity and Excellence in Education*, Vol. 49 No. 4, pp. 480-492, doi: 10.1080/10665684.2016.1227157.

Martin, C. (2015), "Connected learning, libraries, and connecting youth interest", *Journal of Research on Young Adults and Libraries*, available at: www.yalsa.ala.org/jrlya/2015/03/connected-learning-librarians-and-connecting-youth-interest/ (accessed 20 March 2019).

Meyers, E.M. (2016), "Easy as Pi! developing computational thinking in under-served preteens through the public library", paper presented at American Educational Research Association (AERA'16) Conference, 8-12 April, Washington, DC.

Meyers, E.M. (2019), "Secret coders, hidden agendas: debugging children's coding narratives", *Paper to be presented at International Research Society on Children's Literature (IRSCL '19), 13-16 August, Stockholm*.

Meyers, E.M., Erikson, I. and Small, R. (2013), "Digital literacy and informal learning environments: an introduction", *Learning, Media and Technology*, Vol. 38 No. 4, pp. 355-367.

Noble, S. (2018), *Algorithms of Oppression: How Search Engines Reinforce Racism*, NYU Press, New York, NY.

Nodelman, P. (2008), *The Hidden Adult: defining Children's Literature*, Johns Hopkins University Press, Baltimore, MD.

Palfrey, J. and Gasser, E. (2008), *Born Digital: Understanding the First Generation of Digital Natives*, Basic Books, New York, NY.

Papert, S. (1980), *Mindstorms: children, Computers, and Powerful Ideas*, Basic Books, New York, NY.

Prensky, M. (2001), "Digital natives, digital immigrants", *On the Horizon*, Vol. 9 No. 5, pp. 1-6.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K. and Silverman, B. (2009), "Scratch: programming for all", *Communications of the ACM*, Vol. 52 No. 11, pp. 60-67.

Reynolds, R. (2016), "Relationships among tasks, collaborative inquiry processes, inquiry resolutions, and knowledge outcomes in adolescents during guided discovery-based game design in school", *Journal of Information Science*, Vol. 42 No. 1, pp. 35-58.

Reynolds, R. and Harel, C.I. (2011), "Contrasts in student engagement, meaning-making, dislikes, and challenges in a discovery-based program of game design learning", *Educational Technology Research and Development*, Vol. 59 No. 2, pp. 267-289.

Royal Society (2012), "Shut down or restart: the way forward for computing in UK schools", available at: http://royalsociety.org/education/policy/computing-in-schools/report/ (accessed 10 August 2017).

Prato, S.C. (2017), "Beyond the computer age: a best practices intro for implementing library coding programs", *Children and Libraries*, Vol. 15 No. 1, pp. 19-21.

Sheridan, K.M., Halverson, E.R., Litts, B.K., Brahms, L., Jacobs-Priebe, L. and Owens, T. (2014), "Learning in the making: a comparative case study of three makerspaces", *Harvard Educational Review*, Vol. 84 No. 4, pp. 505-556, available at: http://dx.doi.org/10.17763/haer.84.4.brr34733723j648u

Subramaniam, M., Scaff, L., Kawas, S., Hoffman, K.M. and Davis, K. (2018), "Using technology to support equity and inclusion in youth library programming: current practices and future opportunities", *The Library Quarterly*, Vol. 88 No. 4, pp. 315-331.

Susser, D. (2019), "Ethics alone can't fix big tech", *Slate, 7 April*, available at: https://slate.com/technology/2019/04/ethics-board-google-ai.html (accessed 10 May 2019).

Taplin, J. (2017), *Move Fast and Break Things: How Facebook, Google, and Amazon Cornered Culture and Undermined Democracy*, Little, Brown and Co., New York, NY.

Taub, R., Ben-Ari, M. and Armoni, M. (2009), "The effect of CS unplugged on middle-school students' views of CS", *ACM SIGCSE Bulletin*, Vol. 41 No. 3, pp. 99-103.

Toyama, K. (2015), *Geek Heresy: Rescuing Social Change from the Cult of Technology*, Public Affairs Press, New York, NY.

Wagner, T. (2008), *The Global Achievement Gap*, Basic Books, New York, NY.

Wing, J.M. (2008), "Computational thinking and thinking about computing", *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, Vol. 366 No. 1881, pp. 3717-3725.

Wing, J.M. and Stanzione, D. (2016), "Progress in computational thinking, and expanding the HPC community", *Communications of the ACM*, Vol. 59 No. 7, pp. 10-11, available at: https://doi.org/10.1145/2933410

Further reading

Berland, M. and Lee, V.R. (2011), "Collaborative strategic board games as a site for distributed computational thinking", *International Journal of Game-Based Learning*, Vol. 1 No. 2, p. 65.

Denner, J., Werner, L. and Ortiz, E. (2012), "Computer games created by middle school girls: can they be used to measure understanding of computer science concepts?", *Computers and Education*, Vol. 58 No. 1, pp. 240-249.

Halverson, E.R. and Magnifico, A. (2013), "Bidirectional artifact analysis: a method for analyzing digitally-mediated creative processes", in Luckin, R., Puntambekar, S., Goodyear, P., Grabowski, B.L., Underwood, J. and Winters, N. (Eds), *Handbook of Design in Educational Technology*, Taylor and Francis, London and New York, NY.

Wing, J.M. (2006), "Computational thinking", *Communications of the ACM*, Vol. 49 No. 3, pp. 33-35, available at: https://doi.org/10.1145/1118178.1118215