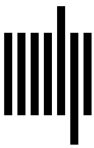
Algorithmic Rights and Protections for Children

Edited by: Mizuko Ito, Remy Cross, Karthik Dinakar, Candice Odgers

Citation:

Algorithmic Rights and Protections for Children
Edited by: Mizuko Ito, Remy Cross, Karthik Dinakar, Candice Odgers
DOI: 10.7551/mitpress/13654.001.0001
ISBN (electronic): 9780262374316
Publisher: The MIT Press
Published: 2023



The MIT Press

6

Designing for Critical Algorithmic Literacies

Sayamindu Dasgupta and Benjamin Mako Hill

Introduction

As pervasive data collection and powerful algorithms increasingly shape children's experiences, children's ability to interrogate computational algorithms is increasingly important. A growing body of work has sought to identify and equip children with the intellectual tools they might use to understand, interrogate, and critique powerful algorithmic systems. We call the intellectual tools that allow children to understand and critique these systems that affect their lives *critical algorithmic literacies*. Unfortunately, because many powerful algorithms are invisible, developing these literacies remains a major challenge. However, it is possible for designers to build systems to support the development of critical algorithmic literacies in children.

Reflecting on extensive observation and design work in the Scratch online community over the last decade, we offer four principles for designers that describe ways to support children in developing critical algorithmic literacies:

1. Enable connections to data
2. Create sandboxes for dangerous ideas

3. Adopt community-centered approaches
4. Support thick authenticity

Our first principle encourages designers to *enable connections to data* by offering children opportunities to engage directly in data analysis, especially with data sets that relate to the world the children live, learn, and play in. The rationale for this principle is that in an increasingly data-driven world, understanding algorithms is deeply connected to understanding data. As children analyze data in order to ask and answer their own questions or pursue their own interests, they create their own algorithms. Through this process, they can start to interrogate both their data and their algorithms.

Our second principle suggests that the development of critical algorithmic literacies can be supported by *creating sandboxes for dangerous ideas*. Algorithms are both powerful and potentially problematic. Our design work suggests that children can develop a deep understanding of both facts when they are allowed to create and experiment with algorithms using carefully designed toolkits. Because these toolkits allow learners to "play with fire" in ways that might lead to negative outcomes, effective toolkit design needs to ensure that the possible dangers are managed and minimized. We use the metaphor of "sandboxes" to describe the goal of managing risk in this design process.

Our third principle suggests that designers should *adopt community-centered approaches* that allow designs to leverage community values that algorithms might challenge. Children belong to many overlapping communities and typically share many of their communities' values. Algorithms are seen as problematic, by children and by society in general, when they violate these socially constituted values. A community-centered approach intentionally situates algorithms within communities that have particular sets of shared values. Doing so makes the problematic nature of algorithms visible to learners who are likely to be aligned with community values that an algorithm violates or challenges.

Finally, we argue that *supporting thick authenticity*—a principle that applies to learning technology design in general—plays a crucial role in developing critical algorithmic literacies. Authenticity in the context of fostering algorithmic or data literacies might mean engaging in activities that involve "real-world" data or scenarios.

First, we describe the theoretical work that informs the way we conceptualize critical algorithmic literacies as well as the empirical and design work we have conducted that has informed our design principles. Next, we describe and situate the four design principles with detailed examples. Finally, we discuss our principles' implications for future design work and conclude with a reflection on unanswered questions and future directions.

Background

Our work draws from the literature on constructionism, a framework for learning and teaching that emphasizes contexts of learning "where the learner is consciously engaged in constructing a public entity, whether it's a sand castle on the beach or a theory of the universe" (Kafai, 2006; Papert & Harel, 1991, p. 1). We are particularly inspired by Resnick and Silverman (2005), who provide a series of design principles for designing constructionist learning environments and toolkits based on reflections on their practice as designers. We have attempted to follow in Resnick and Silverman's footsteps by laying out design principles for critical algorithmic literacies.

We use the term *algorithmic literacies* to describe a subset of computational literacies as articulated by diSessa (2001) in his book *Changing Minds: Computers, Learning, and Literacy*. diSessa suggests three broad pillars for literacy—material, mental or cognitive, and social. Material involves signs, representations, and so on. For language literacy, the material pillar might include alphabets, syntax, and writing conventions. For computational literacies, the material

might involve user interface paradigms like spreadsheets or game genres, or modes of transmission like sharing on social media. The second pillar—mental or cognitive—represents the "coupling" (p. 8) of the material and what goes on inside learners' minds when interacting with the material. The final pillar—social—represents communities that form the basis of literacies. diSessa posits that the emergence of a given literacy is driven by "complex social forces of innovation, adoption, and interdependence" (p. 11).

More recently, Kafai et al. (2019) have proposed a framework with three frames for understanding computational thinking: the cognitive, the situated, and the critical. They call for approaches to computational thinking that integrate "cognitive understanding" in three forms: comprehension of computational concepts; "situated use," meaning that learning happens in contexts the learner cares about; and "critical engagement" to emphasize why we must question the larger structures and processes behind the phenomenon being analyzed. These three frames can also be used in the context of computational literacies. In fact, one of the case studies used by Kafai et al. to illustrate their framework is framed around the concept of "critical data literacies" drawn from our work (Hautea et al., 2017).

Our use of the term "critical" draws from Agre's (2014) idea of "critical technical practice," which ties critique and questioning to the practice of building and creation. In that sense, our goal is not merely knowledge about algorithms (i.e., what algorithms are) but an ability to critique algorithmic systems reflexively. Agre posits critical technical practice as requiring a "split identity—one foot planted in the craft work of design and the other foot planted in the reflexive work of critique" (p. 155). We recognize that as children engage with our toolkits, their design work combined with their reflection allows them to not only understand technical concepts around algorithms (what Agre describes as "esoteric terms") but also evaluate their implications on society ("exoteric terms").

Finally, the notion of critical algorithmic literacies is rooted in Freire's (1986) literacy methods. As we use it, the term was first proposed by Tygel and Kirsch (2016), who noted parallels between Freirean approaches to literacy education and the potential of models for developing data literacy. In suggesting approaches to big data literacy, D'Ignazio and Bhargava (2015) also build on Freire to posit that "[big data] literacy is not just about the acquisition of technical skills but the emancipation achieved through the literacy process" (p. 5). Relatedly, C. H. Lee and Soep (2016) have described their extensive body of work with child-driven multimedia production at the "intersection of engineering and computational thinking on the one hand, and narrative production and critical pedagogy on the other" (p. 481) in terms of *critical computational literacy*. This is a framework developed by C. H. Lee and Garcia (2015) while studying children from south Los Angeles who created animations and interactive games about sociopolitical issues in their community, such as racial profiling.

Our design principles are the result of design and empirical research around two systems we have developed and deployed over the last 10 years: *Scratch Cloud Variables* and *Scratch Community Blocks*. Both tools were designed with constructionist framings of learning in mind. Both sought to support children in learning about computational concepts related to data collection, processing, and analysis. Both tools also built on and extended the Scratch programming language—a widely used, block-based programming language for children (Resnick et al., 2009)—and both were deployed in the Scratch online community, where Scratch users share, comment on, and remix their Scratch projects (Monroy-Hernández & Resnick, 2008).

The primary design goal of Scratch Cloud Variables was to allow children to collect, record, and analyze data within Scratch (Dasgupta, 2013a). The primary goal of Scratch Community Blocks

was to allow children to analyze their own social data directly (Dasgupta & Hill, 2017). Scratch Community Blocks enabled this goal by allowing Scratch users to access and analyze data from the Scratch online community website's database. In deploying both systems, we found that granting children programmatic access to data led them to not only learn the techniques of data analysis but also question and critique data-driven algorithms.

The empirical data that we draw from are from field deployment–based studies we conducted with members of the Scratch online community as well as from face-to-face workshops that we ran in the greater Boston area. For Scratch Cloud Variables, the deployment was part of a larger beta test of the Scratch 2.0 software. For Scratch Community Blocks, 2,500 beta testers were randomly selected from a pool of active Scratch users. Our studies involved observing Scratch projects and comments on projects, as well as seeking feedback through forum posts, surveys, and interviews. To help situate our findings, it is worth noting that the median age of Scratch users is 12 years old, and most are between 11 and 15. Although the distribution varies over time, around two-thirds of Scratch users describe their gender as male, and a small number (approximately 5 percent) do not report gender or self-report using nonbinary genders. Our sample of 2,500 participants in the Scratch Community Blocks was roughly gender balanced but similar in age to the general population of Scratch users (Dasgupta & Hill, 2017, p. 3625).

Design Principles

Over the last decade, much of our research has focused on designing, deploying, and studying systems that seek to support constructionist learning around data and data-intensive algorithms in Scratch. We distill lessons from this work into four principles that

we believe will be useful for a range of designers interested in helping children learn about data-driven computational techniques, as well as question and resist them.

Principle 1: Enable Connections to Data

Our first principle suggests that algorithmic literacies can be supported by offering children opportunities to write programs that interact with data about their worlds. This principle stems from our experience with both Scratch Cloud Variables and Scratch Community Blocks. In both cases, we have found that even relatively simple connections to data from a programming toolkit enable scenarios where children ask questions about the algorithms that shape, store, and use information they create and care about.

We developed Scratch Cloud Variables as part of the second generation of the Scratch programming language (Scratch 2.0). The system allowed Scratch users to store values in variables in ways that persist beyond the run time of their program. Additionally, Scratch Cloud Variables is global in that everyone interacting with a project that uses Scratch Cloud Variables would see the same data (Dasgupta, 2013a). This support for persistent global data, combined with the fact that Scratch 2.0 projects were stored online and ran in a web browser, allowed for functionality in Scratch projects such as global high-score lists, surveys, collaborative art projects, and more (figure 6.1).

During beta testing of the system, children raised concerns about potential privacy threats and control over data made possible by the system. Because a Scratch project could store data persistently, it was possible to create relatively simple Scratch code that would ask for someone's Scratch username (e.g., for a Scratch "guestbook" project) and store it indefinitely. The only way to remove the data, once stored, was to ask the project creator to do so. A Scratch community member noted:

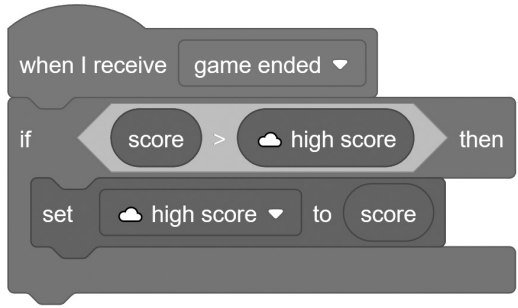


Figure 6.1
An example of a Scratch script using Scratch Cloud Variables. The script determines if the score of the concluded game is higher than the recorded high score. If so, the cloud variable high score (indicated by the cloud icon before the variable name) is updated.

> So if I typed in my first name [into the project] without thinking about it, after that everyone who views the project can see my name [. . .]. Furthermore to remove it I have to contact the owner of the project and request they remove it from the cloud data list. (Dasgupta, 2013b, p. 96)

This example shows that even relatively simple connections with data open up possibilities that enable Scratch community members to think about questions of algorithmic surveillance and power.

We developed the second system, Scratch Community Blocks, by adding programming blocks representing programming primitives into Scratch that could access public metadata about projects and users in the Scratch online community database (Dasgupta & Hill, 2017). An example of the system is shown in figure 6.2. For example, with Scratch Community Blocks, it was possible to create Scratch programs that would access how many times a Scratch project shared in the community had been viewed. Community-wide statistics, such as total number of registered users in the community, were also accessible programmatically through Scratch Community Blocks. These two sets of programming blocks were combined by a young Scratch user to make a project that calculated

Figure 6.2
Example code using Scratch Community Blocks. The code iterates through all the projects shared by the user scratchteam. In each iteration, the Scratch character being controlled by the code says the project title. Image from Dasgupta and Hill (2017).

what proportion of the broader Scratch community had viewed a given Scratch project.

Soon after this project was shared, Scratch community members began discussing the difference between views and viewers (i.e., a single user may view a project multiple times in ways that increase the project's view count). These results were confusing because, at the time, the Scratch website counted views using an algorithm that not only tried to count as many views as possible (e.g., from non-logged-in users) so that project creators would see that their creations had an audience, but also prevented community members from generating views synthetically (e.g., by repeatedly refreshing a project page).[1] Community members noted that one of the most popular projects on the site had a view count that exceeded the number of user accounts on the site.

> **Commenter A:** that's so cool! almost 0.5% of all the users on scratch have viewed my projects and that's a lot :B but crossstitch's[2] results are indeed slightly dubious . . . over 100% of people have viewed his projects which is awesome but impossible—love the project!! ^o^
>
> **Commenter B:** @CommenterA I think it's because its based on views, not each specific player.
>
> **Commenter A:** @CommenterB that's awesome :D people who haven't registered on scratch have viewed a significant amount of his projects yes

> **Project Creator:** @CommenterA Yeah what @CommenterB said is correct.
>
> (Hautea et al., 2017, p. 925)

Through these comments, users worked collectively to show how data are not objective but require *interpretation*, and that data generation is shaped by others' decisions.

Bowker (2005) has argued that "raw data is [. . .] an oxymoron" (p. 184). In an edited volume with the same name, Gitelman (2013) noted that "data are imagined and enunciated against the seamlessness of phenomena" (p. 3). Often, this imagination and enunciation materializes in an algorithm that collects data, such as the viewership statistics of Scratch projects. Enabling children to access that data through algorithms they implemented using Scratch Community Blocks led them to discover illuminating patterns in data, such as view counts of popular projects exceeding the number of community members. As in the extended example of the dialogue about how views are counted on the Scratch website, this in turn led children to attempt to reconstruct how data might have been imagined in the first place.

Resnick and Silverman (2005) posit that "a little bit of programming goes a long way," meaning children can combine relatively simple and limited programming constructs toward a broad range of creative outcomes. In our work, we see a similar phenomenon emerge where simple programming constructs, combined with data in straightforward ways, enable children to uncover structures and assumptions in algorithmic systems. This process allows them to question and discuss algorithmic data collection.

Principle 2: Create Sandboxes for Dangerous Ideas

Our second principle suggests that the development of critical algorithmic literacies can be supported through creating sandboxes for dangerous ideas. Like any sociotechnical tool, algorithms offer

benefits and carry harms. We describe algorithms as "dangerous" to highlight the way they can have powerful, unanticipated, and negative consequences (Smith, 1985). For example, the algorithm behind a real estate search tool may allow the user to filter houses for sale by school rating, but it may not take the history of underfunding of schools in African American and low-income neighborhoods into account. In this way, the search algorithm might unintentionally become a way for potential house buyers to filter for affluent, predominantly White neighborhoods (Noble, 2018, p. 167).

With the deployment of Scratch Community Blocks, metadata about user accounts such as number of followers and number of projects were made programmatically accessible. These numbers can be used as proxies for measures of experience—that is, more projects or more followers indicates more experience with Scratch—but both are far from perfect measures. Although restricting interaction with a project to more experienced users might be an attractive feature to some Scratch users, using these measures as a gatekeeping mechanism can discriminate against newcomers. This was exactly the concern raised by a 13-year-old member of the Scratch community who noted that the algorithm to carry out such discrimination is trivial using Scratch Community Blocks and a single if statement:

> I love these new Scratch Blocks! However I did notice that they could be used to exclude new scratchers or scratchers with not a lot of followers by using a code: like this:
>
> ```
> when flag clicked
>    if then user's followers<300
>       stop all.
> ```
>
> (Hautea et al., 2017, p. 925)

Thus, this young user noted that algorithmic systems can be dangerous in that they can enable discrimination. This type of observation is far from uncommon among Scratch users, and it reflects

a key step toward developing critical algorithmic literacies. That said, it is only possible because of the possibilities introduced by the system.

The notion of engaging with and exploring dangerous ideas is not new in education. Problematic theories are studied as a part of understanding history; discriminatory scenarios are analyzed as a part of engaging with the idea of justice; and potentially physically dangerous experiments are carried out in school and college chemistry laboratories. Although these activities all represent different types of danger, the pedagogical activities around them typically incorporate appropriate safety mechanisms. For the pedagogy of fields like chemistry, this is a topic of ongoing research and study (Alaimo et al., 2010).

An example from our own work that led us to consider the importance of making space for dangerous ideas is a feature introduced in Scratch 2.0—an username block that "reports" the username of the project's viewer if they are logged in (figure 6.3). The username block made new types of functionality possible, including a form of surveillance, by making it much easier to know who (in terms of Scratch usernames) had accessed a given project. As designers, we were also concerned that the block could be used for discrimination within a Scratch project (e.g., by disallowing certain usernames from playing a Scratch game) or to evade moderators (e.g., to have a Scratch project behave in a specific way only for known moderators in the community). On the other hand, we found that the block also made new conversations around surveillance, discrimination, data,

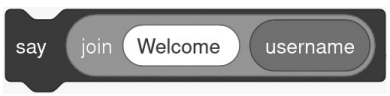


Figure 6.3
The username block introduced in Scratch 2.0. The block "reports" or returns the username of the person viewing the project, or it remains blank if the viewer is not logged in.

and algorithms possible. Achieving a balance between enabling exploration of dangerous ideas and potentially problematic applications of these ideas is not easy. This is especially the case among historically marginalized groups in STEM learning, who may be more vulnerable to discrimination and surveillance.

We only considered the feature because usernames in Scratch are, by community policy, not directly tied to identities in the real world. As a result, the consequences of username-based surveillance in Scratch would be less serious than surveillance of email accounts or other social media accounts. We also considered several approaches to making the username block not report the username directly. Many of these were technical. For example, an initial prototype of the block reported back an alphanumeric value that would remain consistent for a given user accessing a given project over time so that a user interacting with the project could not be identified by username[3] (Dasgupta, 2012). Partly because this idea was difficult to explain, we did not adopt this approach and reverted to the earlier, simpler approach of reporting the username. However, as an added measure, the Scratch project–viewing interface was modified so that it warned users about the block existing in a given project before they ran it and encouraged users to log out of Scratch if they wished to avoid being tracked by a project (Dasgupta, 2013b).

We also deployed the username block with considerable caution, carefully monitored its usage, and were ready to roll back the feature. The Scratch community has a complex and extensive moderation and governance infrastructure that has been described by Lombana Bermúdez (2017) as a combination of "proactive and reactive moderation . . . with the cultivation of socially beneficial norms and a sense of community" (p. 35). We felt that these structures had the potential to prevent and mitigate uses of the feature that could, in theory, go against the friendly nature of the community. In the design phase of the username block, we had many conversations with Scratch's moderation team and with children.

These conversations continued after the feature's launch so we could understand how the new block was being used by the broader community and adapt the system if needed.

To describe our approach of trying to balance exploring "dangerous ideas" and the potentially problematic applications of such ideas, we used the metaphor of sandboxes. Just as a sandbox allows children to play with earth in ways they couldn't for the rest of the playground, our design consists of creating clear boundaries and implementing sociotechnical strategies that prevent any use that might go against community values. The metaphor of a sandbox is common in the field of computer security, where untrusted applications are said to run in a "sandbox" isolated from unneeded resources and other programs (Schreuders et al., 2013). For example, a mobile phone sandboxing system might ensure that an app that does not need access to the camera does not have access to it. In the case of the username block, we spent several design iterations working with children and community moderators to ensure there were enough "walls" (e.g., warning messages in projects that use the block) before we felt we had achieved a balance between encouraging exploration and preventing potential violations of the Scratch community values. In computer and information security pedagogy, using sandboxes to allow learners to experiment with software vulnerabilities is an established practice (Du & Wang, 2008). Computer security researchers and educators have asked students to construct speculative fiction to engage with dangerous ideas and to imagine these ideas' impact on society (Kohno & Johnson, 2011). Our experience suggests that a similar approach may work for critical algorithmic literacies as well.

Principle 3: Adopt Community-Centered Approaches

Our third principle suggests that designers should incorporate community-centered approaches. These approaches would allow a design to leverage existing community values that an algorithm

might change or challenge. This principle reflects increasing recognition of the importance of centering values in computing learning. For example, in a keynote presentation to the 2012 ACM SIGCSE Technical Symposium, Abelson (2012) called for a focus on "computational values," which he defined as commitments that "empower people in the digital world," and which he argued are "central to the flowering of computing as an intellectual endeavor." Justice, respect for privacy, and nondiscrimination are examples of such values.

Prior work in human-computer interaction has argued that values related to a computing system are "something to be discovered" in the context of a given community (Le Dantec et al., 2009, p. 1145). In turn, values can also influence the sense of identity of a learner within their communities. In recent work in the learning sciences, Vakil (2020) proposed the phrase "disciplinary values interpretation" to describe how learners seek to understand what a discipline being studied "is 'all about,' and what it might mean for them to be a part of it as they begin to imagine their future academic, career, and life goals" (p. 7). Vakil has also called for more understanding of, and attention to, "adolescents' political selves and identities, and how these identities become intertwined with learning processes" (p. 22).

In our work, we have seen the dynamic described by Vakil play out as children evaluate technological possibilities in terms of their values. For example, we saw that children using Scratch Community Blocks questioned algorithms by describing algorithmic outcomes that conflicted with the collective value of the Scratch community. Multiple community members expressed concerns about Scratch Community Blocks that enabled projects to rank community members based on the number of followers, and pointed out that this might shift the Scratch community's values from celebrating creativity and expression to emphasizing popularity. For example, a 12-year-old Scratch community member expressed concern that

the new system allowed community members with a smaller number of followers to be made fun of through Scratch projects:

> [. . .] you can easily make fun of someone for example, "You only have 2 followers! Ha! Well I have 10!" (Hautea et al., 2017, p. 927)

Similarly, the aforementioned 13-year-old using Scratch Community Blocks who pointed out that code using the new system could be used to block newcomers from projects thought it was problematic because inclusivity is a core value of the Scratch community, and the algorithmic discrimination that this user correctly identified as being made possible by the system stood in contrast to this value.

One challenge with systems enabling possibilities that go against established community values is that many unsocialized newcomers do not share their new community's values. Zittrain (2006) noted this as a challenge with "generative" systems and platforms, where the outcomes made possible by the system were both positive and negative. With Scratch Cloud Variables, we recognized this issue and implemented a system where the Scratch Cloud Variables feature would only be available to users who were active in the community for some time (Dasgupta, 2013a; Dasgupta & Hill, 2018). By granting only socialized users access to the dangerous feature, we reasoned that newcomers would be allowed to learn Scratch's community values before they could receive access to features that enabled them to flout them.[4]

Our experience suggests that critical approaches to algorithms are driven by the values of the communities in which algorithms are enacted. Of course, communities vary in scope and character, and they can range from groups of friends to families, classrooms, and entire nations. In a 2014 report, the Executive Office of the President invoked values enshrined in the legal structure of the United States when it stated that "big data technologies" have the "potential to eclipse longstanding civil rights protections in how personal

information is used in housing, credit, employment, health, education, and the marketplace" (Podesta, 2014, p. 3).

Of course, not all values are aligned with outcomes that educators seek to reinforce. Values frequently conflict with each other, and widely shared values can sometimes be problematic. Moreover, Benjamin (2019) describes how specific ways of imagining a value—a "master narrative" (p. 134)—can overwhelm alternatives. For example, Philip et al. (2013) draw from their classroom experience in a US public school to describe how the underlying assumption among students debating big data and its implications was "that students, particularly urban children of color, would academically, socially, occupationally, or politically benefit simply by virtue of exposure to big data and new technologies" (p. 117). Philip et al. explain that the design of their new curriculum did not consider the relative lack of opportunities for students of color, leading to an overwhelming focus on one particular framing of big data technology as an equalizer. This example serves as a warning for designers to carefully interrogate a range of values before designing.

Principle 4: Support Thick Authenticity

Finally, we argue that thick authenticity—a principle that applies to learning technology design in general—plays a crucial role in developing critical algorithmic literacies. Authenticity is a complicated concept in the context of learning. Although it is common to encounter terms related to the "real world" and "real-world problems" in popular and scholarly discourse on education, degrees and dimensions of "realness" vary enormously. For example, a learning exercise might involve a fictional scenario where a problem is real but a situation is not (Petraglia, 1998).

Ultimately, "realness" is determined by learners, and "real" learning experiences, problems, and metaphors may be unfamiliar to learners for individual or cultural reasons. Teaching probability using examples based on playing cards may lead to poor learning

experiences if students have never played cards. As corollaries, stronger forms of authenticity emerge when learners have more say in the design and direction of their learning activities, and higher degrees of authenticity are associated with better learning outcomes. For example, while teaching Maori schoolchildren English, Ashton-Warner (1986) found that a compelling strategy to engage her students was to ask them to write about themselves, about their own stories, in their own words—a process she called "organic writing."

Questions of authenticity are likely to be relevant to the type of engagement necessary for supporting the development of algorithmic literacies in children. For example, a baseball analytics algorithm may generate critical engagement when the learner is a young baseball fan who is going to poke holes in the algorithm's assumptions. The same algorithm would likely elicit a lukewarm response, at best, from someone without an interest in baseball. To most learners outside the United States, learning activities that involve baseball are not suitable at all. In our design work, we have drawn inspiration from Ashton-Warner and asked what "organic writing" might look like for developing critical algorithmic literacies (Dasgupta, 2016).

We have also drawn from Shaffer and Resnick (1999), who describe "thick authenticity" as "activities that are personally meaningful, connected to important and interesting aspects of the world beyond the classroom, grounded in a systematic approach to thinking about problems and issues, and which provide for evaluation that is meaningfully related to the topics and methods being studied."

In our work with Scratch Community Blocks, children using the system engaged with complex ideas about power and algorithms because the data they were accessing, and the algorithms they were designing, reflected their experiences, friends, and community in Scratch. If the same interface within Scratch had provided access to nearly any other data source, it would have been less effective in promoting algorithmic literacy among the community of Scratch

users to whom our system was deployed. Considering that most children do not use Scratch, the effectiveness of our systems is likely to be limited among most children.

That said, other contexts might present similarly promising opportunities. For example, families' interactions are increasingly shaped by algorithms and data inherent in smart home technologies. Although it is still less common among children, many people are increasingly collecting data about aspects of their personal lives through quantified-self approaches (Lee, 2013). Because algorithms are increasingly prevalent in a range of contexts, an increasingly wide range of settings offers rich opportunities for promoting algorithmic literacies through thick authenticity.

Discussion

In our own design experiences spanning many iterations, we encountered numerous tensions and open questions in terms of how to best engage the broadest possible set of Scratch community members in critiquing algorithms. The evidence emerging from our work suggests that certain design principles for computational construction kits may support the development of a range of critical algorithmic literacies. Our four design principles reflect a broader perspective—that young learners should go beyond simply observing algorithmic systems and be given opportunities to *create* algorithmic systems. We argue that when children take advantage of these opportunities, some will question algorithms in meaningful ways. In empirical work we conducted, we employed grounded theory (Charmaz, 2006) to analyze the discussions, comments, and activities of children engaging in creative design activities using Scratch Community Blocks (Hautea et al., 2017). Most of the examples we identified of children questioning algorithmic systems emerged from the process of active creation with toolkits.

Though our work is exclusively focused on the Scratch online community, we believe that the lessons we've distilled can be applied to other contexts. For example, the first author of this chapter (Dasgupta) used the "enable connections to data" principle in an introductory college-level Python course to illustrate how gender is often encoded as binary in software. As part of a relatively simple exercise that involved using conditional (if-else) statements, he asked students to use publicly available data on the recommended daily allowance of calcium to make an interactive tool that asked a few questions about an individual (e.g., age, gender) and then recommended a daily allowance of calcium for them.

Because the publicly available data table that is used for this purpose encoded gender as binary,[5] students ended up designing their programs with the inbuilt assumption of binary gender, with a small proportion of the students questioning the practice in their submission (e.g., as code comments). This provided Dasgupta with an opportunity to engage students—after they had written the program—in a discussion starting with the prompt, "What's wrong with this assignment?" Students pointed to the notion of a binary variable to represent gender, and this led to a broader discussion about the choices programmers make in modeling the world in their algorithms (Costanza-Chock, 2018; Smith, 1985).

Similarly, one might use the principle of "sandboxes for dangerous ideas" in a web programming exercise for college students by scaffolding an exercise around a survey system's design and conversations around the choices students make about tracking their survey-participants' identities. What is an acceptable technical solution to prevent repeat participation in an anonymous survey? Can a "technical solution" even exist? There are likely many other ways to help children understand and question algorithmic systems. For example, approaches such as co-designed games have yielded promising results for engaging children in understanding privacy (Kumar et al., 2018). Similar approaches may emerge

toward other aspects of critiquing data and algorithm-driven systems as well.

Limitations

Our work is limited in that it has focused on individual learners and case studies. Our examples are also limited in that they show what is possible when our design implications are embraced, but not necessarily what is likely. For example, Scratch users—especially those who chose to engage with us by responding to our recruitment calls, coming to our workshops, and participating in other ways—are unlikely to represent children in general. We do not claim that the examples described here represent how most children would respond to the same tools and contexts. Indeed, we believe that the uniqueness of children's context, interests, and backgrounds likely means that no single tool or approach will work for all children.

Moreover, although several of our study participants questioned and critiqued algorithmic possibilities, many engaged with at least some of these problematic possibilities uncritically (e.g., by making Scratch Community Blocks projects that would only work for community members with more than five followers). More structured support, such as the "what's wrong with this assignment?" prompt in the classroom example we provided, will often be needed to engage a broader group in considering these questions.

In a sense, these facts remind designers and educators of the perils of a technocentric approach in learning—"the tendency to give . . . centrality to a technical object" (Papert, 1987, p. 23). Brennan (2015) offered a model of how a designer can support the work of educators toward using technology in the service of learning in a nontechnocentric way. Beyond systems and curriculum designed based on the four principles we have outlined, such models represent

essential and complementary components that need to be in place in a learning environment for learners to develop critical algorithmic literacies. By stating these limitations and recognizing that this is a work in progress, we offer our principles in the hopes that other designers of educational technologies and experiences will build on our work and contribute to the larger project of fostering critical algorithmic literacies in children.

Conclusion

In their paper on designing construction kits for children, Resnick and Silverman (2005) present their final design principle as "iterate, iterate—then iterate again." They conclude by stating that this applies to their principles as well. Our four principles are no exceptions to this excellent advice. We intend to keep iterating on our principles, taking them apart, putting them back together, and changing them. We offer our principles with humility and a sincere desire to work toward the dual goals of supporting children in understanding the algorithmic systems that are increasingly shaping their worlds and doing what we can to give them the intellectual tools to question and resist them.

Acknowledgments

Many of the projects referred to in this chapter were financially supported by the US National Science Foundation (1002713, 1027848, DRL-1417663, and DRL-1417952). We would also like to acknowledge feedback and support from Mitchel Resnick, Natalie Rusk, Hal Abelson, Brian Silverman, Amos Blanton, and other members of the Scratch team. Finally, none of this work would have been possible without the children who generously tried out our

new technologies, gave us feedback, and inspired us in multiple ways with their ingenuity and kindness.

Notes

1. The first author of this article had implemented this particular algorithm for the Scratch website at that time.

2. crossstitch (username changed) was at that time the creator of some of the most popular projects on Scratch.

3. Internally, this alphanumeric value was being generated by one-way cryptographic hash of the username, project ID, and a cryptographic salt.

4. A second part of the reasoning is that if someone used Scratch Cloud Variables in a problematic way, they might get banned and lose access to the account into which they had poured substantial time and resources.

5. Current versions of the table and the associated text have discontinued the practice; earlier versions of the page can be accessed from the Internet Archive (e.g., https://web.archive.org/web/20200817125236/https://ods.od.nih.gov/factsheets/Calcium-HealthProfessional/).

References

Abelson, H. (2012). From computational thinking to computational values. *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education*, 239–240. https://doi.org/10.1145/2157136.2157206

Agre, P. E. (2014). Toward a critical technical practice: Lessons learned in trying to reform AI. In G. Bowker, S. L. Star, L. Gasser, & W. Turner (Eds.), *Social science, technical systems, and cooperative work: Beyond the great divide* (pp. 131–157). Taylor & Francis. http://public.ebookcentral.proquest.com/choice/publicfullrecord.aspx?p=1689049

Alaimo, P. J., Langenhan, J. M., Tanner, M. J., & Ferrenberg, S. M. (2010). Safety teams: An approach to engage students in laboratory safety. *Journal of Chemical Education*, *87*(8), 856–861. https://doi.org/10.1021/ed100207d

Ashton-Warner, S. (1986). *Teacher*. Simon & Schuster.

Benjamin, R. (2019). *Race after technology: Abolitionist tools for the new Jim code*. Polity.

Bowker, G. C. (2005). *Memory practices in the sciences*. MIT Press.

Brennan, K. (2015). Beyond technocentrism: Supporting constructionism in the classroom. *Constructivist Foundations*, *10*(3), 289–296. http://constructivist.info/10/3/289.brennan

Charmaz, K. (2006). *Constructing grounded theory: A practical guide through qualitative analysis*. Sage Publications.

Costanza-Chock, S. (2018, July 16). Design justice, A.I., and escape from the matrix of domination. *Journal of Design and Science*. https://doi.org/10.21428/96c8d426

Dasgupta, S. (2012). *Learning with data: A toolkit to democratize the computational exploration of data* [Masters thesis, Massachusetts Institute of Technology]. DSpace@MIT. http://dspace.mit.edu/handle/1721.1/78203

Dasgupta, S. (2013a). From surveys to collaborative art: Enabling children to program with online data. *Proceedings of the 12th International Conference on Interaction Design and Children (IDC '13)*, 28–35. https://doi.org/10.1145/2485760.2485784

Dasgupta, S. (2013b). Surveys, collaborative art and virtual currencies: Children programming with online data. *International Journal of Child-Computer Interaction*, *1*(3–4), 88–98. https://doi.org/10.1016/j.ijcci.2014.02.003

Dasgupta, S. (2016). *Children as data scientists: Explorations in creating, thinking, and learning with data* [Doctoral thesis, Massachusetts Institute of Technology]. DSpace@MIT. http://dspace.mit.edu/handle/1721.1/107580

Dasgupta, S., & Hill, B. M. (2017). Scratch community blocks: Supporting children as data scientists. *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17)*, 3620–3631. https://doi.org/10.1145/3025453.3025847

Dasgupta, S., & Hill, B. M. (2018). How "wide walls" can increase engagement: Evidence from a natural experiment in Scratch. *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18)*, 361:1–361:11. https://doi.org/10.1145/3173574.3173935

D'Ignazio, C., & Bhargava, R. (2015, September). Approaches to building big data literacy. *Proceedings of the Bloomberg Data for Good Exchange Conference 2015*. https://dam-prod2.media.mit.edu/x/2016/10/20/Edu_D'Ignazio_52.pdf

diSessa, A. A. (2001). *Changing minds: Computers, learning, and literacy*. MIT Press.

Du, W., & Wang, R. (2008). SEED: A suite of instructional laboratories for computer security education. *Journal on Educational Resources in Computing*, *8*(1), 1–24. https://doi.org/10.1145/1348713.1348716

Freire, P. (1986). *Pedagogy of the oppressed*. Continuum.

Gitelman, L. (Ed.). (2013). *"Raw data" is an oxymoron*. The MIT Press.

Hautea, S., Dasgupta, S., & Hill, B. M. (2017). Youth perspectives on critical data literacies. *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17)*, 919–930. https://doi.org/10.1145/3025453.3025823

Kafai, Y. (2006). Constructionism. In K. R. Sawyer (Ed.), *The Cambridge handbook of the learning sciences* (1st ed., pp. 35–46). Cambridge University Press.

Kafai, Y., Proctor, C., & Lui, D. (2019). From theory bias to theory dialogue: Embracing cognitive, situated, and critical framings of computational thinking in K-12 CS education. *Proceedings of the 2019 ACM Conference on International Computing Education Research*, 101–109. https://doi.org/10.1145/3291279.3339400

Kohno, T., & Johnson, B. D. (2011). Science fiction prototyping and security education: Cultivating contextual and societal thinking in computer security education and beyond. *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education—SIGCSE '11*, 9. https://doi.org/10.1145/1953163.1953173

Kumar, P., Vitak, J., Chetty, M., Clegg, T. L., Yang, J., McNally, B., & Bonsignore, E. (2018). Co-designing online privacy-related games and stories with children. *Proceedings of the 17th ACM Conference on Interaction Design and Children—IDC '18*, 67–79. https://doi.org/10.1145/3202185.3202735

Le Dantec, C. A., Poole, E. S., & Wyche, S. P. (2009). Values as lived experience: Evolving value sensitive design in support of value discovery. *Proceedings of the 27th International Conference on Human Factors in Computing Systems—CHI '09*, 1141. https://doi.org/10.1145/1518701.1518875

Lee, C. H., & Garcia, A. D. (2015). "I want them to feel the fear . . .": Critical computational literacy as the new multimodal composition. In *Gamification: Concepts, methodologies, tools, and applications* (pp. 2196–2211). IGI Global. https://doi.org/10.4018/978-1-4666-8200-9.ch111

Lee, C. H., & Soep, E. (2016). None but ourselves can free our minds: Critical computational literacy as a pedagogy of resistance. *Equity & Excellence in Education*, *49*(4), 480–492. https://doi.org/10.1080/10665684.2016.1227157

Lee, V. R. (2013). The quantified self (QS) movement and some emerging opportunities for the educational technology field. *Educational Technology*, *53*(6), 39–42. http://www.jstor.org/stable/44430216

Lombana Bermúdez, A. (2017). Moderation and sense of community in a youth-oriented online platform: Scratch's governance strategy for addressing harmful speech. In *Perspectives on Harmful Speech Online*. Berkman Klein Center for Internet & Society Research Publication. https://dash.harvard.edu/handle/1/33746096

Monroy-Hernández, A., & Resnick, M. (2008). Empowering kids to create and share programmable media. *Interactions*, *15*(2), 50–53. https://doi.org/10.1145/1340961.1340974

Noble, S. U. (2018). *Algorithms of oppression: How search engines reinforce racism*. NYU Press.

Papert, S. (1987). Computer criticism vs. technocentric thinking. *Educational Researcher*, *16*(1), 22–30. https://doi.org/10.3102/0013189X016001022

Papert, S., & Harel, I. (1991). Situating constructionism. In *Constructionism* (Vol. 36, pp. 1–11). Ablex Publishing.

Petraglia, J. (1998). *Reality by design: The rhetoric and technology of authenticity in education*. Lawrence Erlbaum Associates. http://www.myilibrary.com?id=232329

Philip, T. M., Schuler-Brown, S., & Way, W. (2013). A framework for learning about big data with mobile technologies for democratic participation: Possibilities, limitations, and unanticipated obstacles. *Technology, Knowledge and Learning*, *18*(3), 103–120. https://doi.org/10.1007/s10758-013-9202-4

Podesta, J. (2014). *Big data: Seizing opportunities, preserving values*. Executive Office of the President.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., & Kafai, Y. (2009). Scratch: Programming for all. *Communications of the ACM*, *52*(11), 60–67. https://doi.org/10.1145/1592761.1592779

Resnick, M., & Silverman, B. (2005). Some reflections on designing construction kits for kids. *Proceedings of the 2005 Conference on Interaction Design and Children*, 117–122. https://doi.org/10.1145/1109540.1109556

Schreuders, Z. C., McGill, T., & Payne, C. (2013). The state of the art of application restrictions and sandboxes: A survey of application-oriented access controls and their shortfalls. *Computers & Security*, *32*, 219–241. https://doi.org/10.1016/j.cose.2012.09.007

Shaffer, D. W., & Resnick, M. (1999). "Thick" authenticity: New media and authentic learning. *Journal of Interactive Learning Research*, *10*(2), 195–215.

Smith, B. C. (1985). The limits of correctness. *ACM SIGCAS Computers and Society*, *14*, *15*(1, 2, 3, 4), 18–26. https://doi.org/10.1145/379486.379512

Tygel, A., & Kirsch, R. (2016). Contributions of Paulo Freire for a critical data literacy: A popular education approach. *The Journal of Community Informatics*, *12*(3). https://ojs.uwaterloo.ca/index.php/JoCI/article/view/3279

Vakil, S. (2020). "I've always been scared that someday I'm going to sell out": Exploring the relationship between political identity and learning in computer science education. *Cognition and Instruction*, *38*(2), 87–115. https://doi.org/10.1080/07370008.2020.1730374

Zittrain, J. L. (2006). The generative internet. *Harvard Law Review*, *119*(7), 1974–2040. http://www.jstor.org/stable/4093608

The open access edition of this book was made possible by generous funding from the MIT Libraries.

The MIT Press would like to thank the anonymous peer reviewers who provided comments on drafts of this book. The generous work of academic experts is essential for establishing the authority and quality of our publications. We acknowledge with gratitude the contributions of these otherwise uncredited readers.

This book was set in Stone Serif and Stone Sans by Westchester Publishing Services.

Library of Congress Cataloging-in-Publication Data

Names: Ito, Mizuko editor. | Cross, Remy, editor. | Dinakar, Karthik, editor. | Odgers, Candice L., editor.
Title: Algorithmic rights and protections for children / edited by Mizuko Ito, Remy Cross, Karthik Dinakar, Candice Odgers.
Description: Cambridge, Massachusetts : The MIT Press, 2023. | Includes bibliographical references and index.
Identifiers: LCCN 2022034631 (print) | LCCN 2022034632 (ebook) | ISBN 9780262545488 (paperback) | ISBN 9780262374323 (epub) | ISBN 9780262374316 (pdf)
Subjects: LCSH: Computers and children—United States. | Internet and children—United States. | Computer algorithms—Social aspects—United States. | Data privacy—United States. | Computer literacy—United States. | Internet—Safety measures. | Children's rights—United States.
Classification: LCC QA76.9.C659 A44 2023 (print) | LCC QA76.9.C659 (ebook) | DDC 004.083—dc23/eng/20221108
LC record available at https://lccn.loc.gov/2022034631
LC ebook record available at https://lccn.loc.gov/2022034632